ARO 23726.1-EG

(2)

ANALYSIS OF A SYSTEM TO PREVENT HELICOPTER ROTOR BLADE-AIRFRAME STRIKES

Final Report

B. W. McCormick
R. G. Melton

May 1, 1989

U. S. ARMY RESEARCH OFFICE

Contract/Grant Number DAAL03-87-K-0002

Department of Aerospace Engineering
The Pennsylvania State University
University Park, PA 16802

UNCLASSIFIED MASTER COPY - FOR REPRODUCTION PURPOSES

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Pennsylvania State University 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION U. S. Army Research Office
6c. ADDRESS (City, State, and ZIP Code) Department of Aerospace Engineering 233 Hammond Bldg University Park, PA 16802	7b. ADDRESS (City, State, and ZIP Code) P. O. Box 12211 Research Triangle Park, NC 27709-2211
8a. NAME OF FUNDING/SPONSORING ORGANIZATION U. S. Army Research Office 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) P. O. Box 12211 Research Triangle Park, NC 27709-2211	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

Analysis of a System to Prevent Helicopter Rotor Blade-Airframe Strikes (Unclassified)

12. PERSONAL AUTHOR(S) B. W. McCormick, R. G. Melton

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 86-11-1 TO 88-12-31	89-04-03	111

16. SUPPLEMENTARY NOTATION The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

Helicopter rotor stability, rotor flapping, rotor dynamics, rotor control systems

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Rotor blade-airframe strikes are rare but they do occur. Three areas of the airframe are particularly vulnerable: the tail boom, canopy and, in the case of the underslung, teetoring rotor, the rotor shaft. This latter case is known as mast bumping. This report studies a system to prevent a helicopter rotor blade from striking any part of the airframe. Essentially, the system continuously predicts ahead the rotor blade flapping in response to an input such as pilot control or an atmospheric disturbance. If a blade strike is predicted to occur then an

(cont'd. on reverse)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

A

(19., cont'd.)

appropriate feedback control is applied to alter the future flapping. The prediction is then begun again with the altered control. In the actual system, an enunciator might warn the pilot at the time that he is attempting a control input which could be hazardous. Two somewhat independent approaches to the design of the controller are taken. One of the programs is entirely numerical in its approach. The other ~~utilizes~~ uses modern control theory and considers the preliminary aspects of implementing the controller in digital hardware. Both methods indicate the feasibility of preventing excessive flapping, although the question of implementation in a dedicated microprocessor is not fully resolved. [illegible]

Table of Contents

Analysis of a System to Prevent Helicopter Rotor Blade-Airframe Strikes

B. W. McCormick[1] and R G. Melton[2]

Abstract

Rotor blade-airframe strikes are rare but they do occur. Three areas of the airframe are particularly vulnerable: the tail boom, canopy and, in the case of the underslung, teetoring rotor, the rotor shaft. This latter case is known as mast bumping. This report studies a system to prevent a helicopter rotor blade from striking any part of the airframe. Essentially, the system continuously predicts ahead the rotor blade flapping in response to an input such as pilot control or an atmospheric disturbance. If a blade strike is predicted to occur then an appropriate feedback control is applied to alter the future flapping. The prediction is then begun again with the altered control. In the actual system, an enunciator might warn the pilot at the time that he is attempting a control input which could be hazardous. Two somewhat independent approaches to the design of the controller are taken. One of the programs is entirely numerical in its approach. The other utilizes modern control theory and considers the preliminary aspects of implementing the controller in digital hardware. Both methods indicate the feasibility of preventing excessive flapping, although the question of implementation in a dedicated microprocessor is not fully resolved.

Nomenclature

The nomenclature is to be found following the text. It includes both symbols used in the text as well as those used in the computer codes.

Introduction

The purpose of this study is to determine the feasibility of a control system which will prevent a helicopter rotor blade from striking any part of the airframe. Essentially, the idea of the system is to continually predict ahead how the rotor blade will flap in response to an input to the rotor such as pilot control or an atmospheric disturbance. If a blade strike is predicted to occur then an appropriate feedback control is applied to alter the future flapping. The prediction is then begun again with the altered control. In the actual system, an enunciator might warn the pilot at the time that he is attempting a control input which could be hazardous.

[1]Boeing Professor of Aerospace Engineering

[2]Associate Professor of Aerospace Engineering

Rotor blade-airframe strikes have occurred on several helicopter configurations. Three areas of the airframe are particular vulnerable, the tail boom, canopy and, in the case of the underslung, teetoring rotor, the rotor shaft. This latter case, known as "mast bumping", was the primary motivation for this study but the results should be generally applicable to other helicopter configurations.

Two somewhat independent approaches to the design of the controller have been taken. One of the programs is entirely numerical in its approach. The other utilizes modern control theory. Both approaches require that the following questions be answered.

1. How many revolutions of the rotor must the flapping be predicted ahead?
2. What type of feedback control is required?
3. How quickly, in terms of a rotor revolution, must the microprocessor predict the future flapping?

The answers to these questions must depend, in part, upon the operating state of the rotor, the point around the azimuth where the strike is predicted to occur, and the severity of the predicted strike.

The material to be developed here attempts to answer the above questions but with certain limitations. It proved to be a more difficult task than had been anticipated so that, with the allotted funding, it has not been possible to include, as yet, the dynamics of the airframe, nor to determine, with any certainty, whether or not a microprocessor can provide the speed which is necessary. Based on informal discussions with persons knowledgeable in the design of microprocessors, it is felt at this time that the necessary speed can be achieved. Also, it may be possible, within the accuracy required, to replace some of the numerical integrations with approximate closed-form expressions, thus increasing the speed of the calculations.

This report begins by presenting the development of a non-linear, numerical program to predict the flapping of a rotor including retreating blade stall and reversed flow. This program is utilized in both the numerical study and in the one utilizing modern control theory. Next, the logic for the numerical controller is presented together with some results which were obtained using the AH-1J helicopter as an example. This is followed by the analytical developments based on modern control theory and some results of that analysis, again using the AH-1J as an example. Finally, some conclusions and recommendations are made, the principal one being that the scheme for preventing blade strikes appears to be feasible and should be pursued further.

Description of Program to Predict Rotor Blade Flapping

A program to predict blade flapping was written specifically for this effort for two reasons. First, it was uncertain that a classical approach which, at any azimuth position, obtains the integrated blade lift and hub moment in closed form, would be adequate. The classical approach is limited to first harmonic flapping and contains certain small angle assumptions which may be significant. The classical approach also neglects reverse flow and retreating blade stall. Secondly, at the other extreme, it was felt that the computational time required by existing elaborate codes which predict rotor flapping, such as C-81, would be prohibitive for the proposed control system. Thus a compromise between these two extremes was taken.

The subroutine which was written begins with the rotor state at a particular instant and azimuth angle, ψ, and integrates the thrust and torque over the radius. A uniform downwash is assumed together with a rotor blade which is rigid but semi-articulated with only a flapping degree of freedom. A better model of the downwash, like a triangular variation or a prescribed wake, could be incorporated into this model but consideration of a flexible blade would require extensive modification. At every azimuth position, the blade loading is numerically integrated along the span to obtain the instantaneous lift and hub moment. No small angle assumptions are made with regard to the angle of attack of the blade sections. Reverse flow and stall are accounted for by the use of a table lookup to obtain the airfoil lift and drag coefficients for angles of attack from zero to 360 degrees.

For completeness, the analytical basis for the flapping program will be presented in detail. Consider figure 1 which is a left side view of a rotor with the disc plane at an angle of attack. The disc plane is sometimes referred to as the shaft plane and is the plane normal to the shaft. From this figure, it is seen that the freestream velocity, V, can be split into two components, one normal to the disc plane and the other lying in the plane.

Now consider figure 2. This figure is a top view of the disc with the blade at an azimuth angle, ψ, measured clockwise from the downstream position. It is seen that the in-plane velocity can be further divided into two components, one parallel and the other normal to the blade. Also shown in this figure is the linear velocity component directed normal to the blade at a radius of r which results from the angular velocity of the rotor.

Figure 3 is a view in the plane defined by the blade and the shaft axis. The blade is shown with a flapping angle, β, a flapping velocity, $d\beta/dt$ and an angular acceleration about the flapping axis. At a radius of r, if the blade is flapping up, as shown relative to the blade, a downward velocity results from the upward flapping. Further, the previous component of the velocity parallel to the rotor has, itself, a component directed up normal to the blade.

Figure 4 shows the blade section at the radius, r, at a pitch angle Θ relative to the disc plane. The net velocity up, normal to the disc plane, is given by,

$$V_u = V \sin \alpha_s - w - (r-\epsilon)\dot{\beta} - \beta V \cos \alpha_s \cos \psi \qquad (1)$$

Where:

w = rotor downwash velocity
r = radial distance of blade section from shaft axis
ϵ = radial distance of flapping hinge from shaft axis
ψ = azimuth angle
V = velocity at which rotor is advancing
β = flapping angle between rotor blade and disc plane
α_s = disc plane angle of attack relative to V

The net velocity in the disc plane can be obtained from,

$$V_t = \omega r + V\cos \alpha_s \sin \psi \qquad (2)$$

Where:

ω = angular velocity of rotor

Thus, from figure 4 and equations (1) and (2), the angle of attack of the blade section is given by,

$$\alpha = \Theta + \phi \tag{3}$$

Where:

$$\phi = \tan^{-1}(V_t/V_u)$$

One must be careful in programming the above to remember that the angle ϕ can lie in any quadrant and can be of a magnitude such that the angle of attack, α, of the section can vary from 0 to 360 degrees. In order to obtain the correct section C_l and C_d, and to correctly resolve the lift and drag forces into the thrust and torque directions, one should check the sign of both V_t and V_u. Figure 5 illustrates the possible flow directions and the corresponding lift and drag vectors. In the subroutine SUBFLAP, which is appended to this report and which will be discussed in more detail later, in lieu of using equation (3) to calculate the angle of attack, the angle ϕ is first calculated simply as:

$$\phi = \tan^{-1}\frac{|V_t|}{|V_u|} \tag{4}$$

The angle of attack is then found by the four possible combinations shown in figure 5. Specifically,

$$V_t > 0 \text{ and } V_u > 0 \qquad \alpha = \Theta + \phi \tag{5a}$$
$$V_t > 0 \text{ and } V_u < 0 \qquad \alpha = \Theta - \phi \tag{5b}$$
$$V_t < 0 \text{ and } V_u > 0 \qquad \alpha = \pi + \Theta - \phi \tag{5c}$$
$$V_t < 0 \text{ and } V_u > 0 \qquad \alpha = \pi - \Theta - \phi \tag{5d}$$

In the normal case, the lift coefficient, C_l, adds to the rotor thrust and to the torque while the drag coefficient, C_d, adds to the torque and subtracts from the thrust. However, in the reverse flow region, or anywhere along the blade where the angle of attack is greater than 90 degrees, this is no longer the case. Thus factors multiplying C_l and C_d are set equal to 1 or -1 depending upon whether or not the resultant flow is impinging on the leading edge or trailing edge and from above or below.

Data for airfoils over an alpha range from 0 to 360 degrees is difficult to find so that, for this study, the data for the NACA 0012 airfoil from 0 to 180 degrees was used. This is the same data used in C-81. The logic for determining C_l, C_d and the factors for resolving the lift is found, and identified, in the appended subroutine.

Knowing C_l and C_d from a table lookup and the angles above, the derivatives of the rotor lift and drag can be found from,

$$\frac{dL}{dr} = \frac{1}{2}\rho V_e^2 c C_l \tag{6a}$$
$$\frac{dD}{dr} = \frac{1}{2}\rho V_e^2 c C_l \tag{6b}$$

where V_e is the resultant velocity shown in figure 4 and given by,

$$V_e = (V_u^2 + V_t^2)^{1/2} \tag{7}$$

The radial derivative of the blade thrust is then found from,

$$\frac{dT}{dr} = \frac{dL}{dr}\frac{|V_t|}{V_e} + \frac{dD}{dr}\frac{|V_u|}{V_e} \tag{8}$$

The derivative of the moment about the flapping hinge is found by multiplying the above by the distance, $r-\epsilon$, from the flapping hinge to the blade element. Thus,

$$\frac{dM}{dr} = (r - \epsilon)\frac{dT}{dr} \tag{9}$$

Using a simple trapezoidal rule, equations (8) and (9) are integrated along the blade from ϵ to R to obtain the total instantaneous thrust and hinge moment. Knowing the blade moment of inertia and the moment of the blade weight about the flapping hinge, the angular acceleration about the flapping hinge is then calculated from,

$$I_F\frac{d^2\beta}{dt^2} = M - I_F\,\beta\,w^2 - M_W \tag{10}$$

The angular velocity and the flapping angle, β, at the end of the time increment, Δ t, are then obtained from,

$$\frac{d\beta}{dt}(t + \Delta\ t) = \frac{d\beta}{dt}(t) + \frac{d^2\beta}{dt^2}\Delta t \tag{11}$$

$$\beta\,(t + \Delta t) = \beta(t) + \frac{d\beta}{dt}\Delta t + \frac{d^2\beta}{dt^2}\frac{\Delta t^2}{2} \tag{12}$$

This is essentially the end of the flapping subroutine. The new state of the rotor defined by (11) and (12) and the instantaneous thrust from (8) is returned to the main program to be integrated with respect to the azimuth angle, ψ, and to be used in the logic of the controller.

Numerical Controller

The action of the controller is shown schematically in figure 6. Here, the flapping angle, β, is shown as a function of time. Suppose, for example, that the controller is activated at the time corresponding to point A. At that time it begins to predict the flapping of the rotor ahead for a specified number of revolutions based on the state of the rotor at the time and on a linear extrapolation of the control input. If the rotor flapping is predicted to be within limits, the controller returns to the rotor after a time, τ, which is the time required to perform the calculations. During this time the rotor blade has moved from A to point B. The process is then repeated. As illustrated, the flapping at point A was predicted to be within prescribed limits for three revolutions ahead. Thus the controller simply returns after the time τ to sense a new rotor state in order to perform another prediction.

Suppose, at some later time, point C, the controller begins a prediction during which, because of a control input, the rotor flapping is predicted to exceed a limiting value. Upon reaching this limit the prediction immediately stops and the controller returns to the rotor at some fraction of τ, point D. At this instant it commands an incremental step input of corrective control to prevent the excessive flapping which was predicted. It then begins a new prediction with the incremental control. If excessive flapping is not predicted then an incremental step of corrective control is removed. Thus, the control actuators commanded by the controller are generally inactive except for those rare occasions when excessive control is applied by the pilot or when external disturbances are sufficient to cause excessive flapping.

The logic for the above is shown in figure 7. This logic serves several purposes. First, it calls for the subroutine FLAP in order to model the operating rotor. Secondly it calls for this same subroutine to model the action of predicting the future flapping of the rotor and finally, it checks the future flapping against prescribed limits and provides the necessary feedback control to alter the predicted future flapping. The subroutine DNWSH provides the downwash velocity as a function of blade azimuth position and rotor thrust. To date, as stated previously, only a simple uniform downwash model has been used. The subroutine CNTRL provides the corrective control which is a function of how excessive the flapping is predicted to be and the azimuth position at which it is predicted to occur.

Figure 7 will now be explained in some detail. The algorithm begins by inputting the rotor state, operating conditions and parameters defining the controller. The state of the rotor; ie, the advance velocity, trim angle of attack and control angles are read from data files with the angles being determined from the static trim program described in reference (1). Parameters governing the rotor flapping, namely the rotor geometry and inertia properties, are also read in from data files. The variables specific to a particular numerical calculation are read in from the keyboard by the operator. Specifically, in order, these are:

1. Identifying case number
2. Maximum flapping angle to be allowed, BETLIM
3. Increments to cyclic pitches if allowable flapping is predicted to be exceeded, DELFB1 and DELFB2
4. Maximum feedback on cyclic controls, FB1LIM and FB2LIM
5. Number of rotations before control input (to disturb static trim) is applied, N
6. Number of rotations that flapping is to be predicted ahead at a given instant, NPRED
7. Fraction of a revolution required to accomplish the above prediction, FPRED
8. Rate of linear increase of cyclic control input to disturb system, CRATE1 and CRATE2
9. Maximum incremental values for the cyclic control inputs, THE1LM and THE2LM
10. Length of time for cyclic control inputs to be applied, TOFF
11. Number of numerical integrations between printouts, PRT
12. Maximum number of revolutions for run, NMAX

Following the input, the time, azimuth angle and quantities to be integrated are initialized to either zero or to their initial values. The switch KNTRL is set to 1 and the calculations begin. The circled nodes in Figure 7 numbered 1 through 21 refer to corresponding statement numbers in the program FLAP. The value of KNTRL remains at 1 until the time is reached for the control input to begin. At any instant of time and azimuth angle, ψ, the subroutine SUBFLAP integrates the blade thrust with respect to radius in order to obtain the derivative of the total rotor thrust with respect to ψ.

Following statement 5, ψ is checked to see if it exceeds the value of 2π or the value at which control input begins. If so, KNTRL is set to 2 and, after saving the current state variables as initial values for later use, the prediction of the future rotor flapping begins.

The prediction continues over NPRED revolutions or until the predicted flapping exceeds BETLIM. During all of the calculations, after every revolution, the thrust is averaged and the subroutine DWNWSH called to update the average downwash velocity. The time, TC, required for the predictions is calculated following statement 14, correspondent to the time at which BETLIM was exceeded, or simply equated to the time required for NPRED revolutions if BETLIM was not exceeded. The program then sets KNTRL equal to 3 and returns to the initial conditions saved when the prediction started. Calculations of the flapping are then repeated until the time, TC, is reached. At this point, the parameter FB is increased by 1 which adds an increment of feedback control. If it is predicted that BETLIM will not be exceeded, then FB is reduced by 1. However, the subroutine SUBCNTRL does not allow FB to be less than zero. At this time, and with the feedback, KNTRL is set again to zero and the prediction begins again. This alternating process of predicting the future flapping and calculating the actual flapping with feedback continues until NMAX revolution are reached.

Results from the Numerical Program

The AH-1J helicopter was used as an example to demonstrate the programs developed here with numerical values of the parameters for this helicopter and its rotor system being obtained from reference (2).

Comparison with Predictions from Classical Theory

To begin, a comparison was made between predictions of steady flapping based on the numerical code with those based on classical theory. (see reference 3) The typical results shown in Figures 8 and 9 appear to confirm the code and offer the promise of being able to use the linearized approximations contained in the classical formulation to speed up the numerical controller. For these particular operating states, the amplitude of the flapping, as predicted by the classical theory, agrees almost exactly with the predictions from the numerical model. These particular graphs are for an AH-1J operating at 61 knots at 3075 feet at a gross weight of 9500 lbs. In figure 8, the controls are held constant. In Figure 9, the lateral cyclic is increased rapidly by 10 degrees after two revolutions and then held constant.

Rotor Response

The response of the rotor to a lateral control input is presented in Figure 10. Here, the helicopter is trimmed at 80 kts at standard sea level conditions using the static trim program in reference 1. The rotor revolutions are measured starting with the blade at an azimuth angle of zero. Since this latter program, based on classical, linearized theory differs slightly from the numerical predictions of flapping, it takes approximately two revolutions for the numerical results to become steady. At 3.0 and at 3.5 revolutions, a step increment in the lateral cyclic control input of 5 degrees is applied. It can be seen that the flapping responds rapidly to the control input and reaches a steady state in approximately one and one-half revolutions or less. When the lateral cyclic control is applied at an azimuth angle of zero degrees (3 revolutions), the lateral flapping is seen to increase by approximately two degrees only a quarter of a revolution later at a psi value of 90 degrees, and by three and one-half degrees another half of a revolution later at a psi of 270 degrees.

The effect of delaying a correction to a disturbance is illustrated by Figures 11 and 12. In both figures, a lateral cyclic control of 5 degrees is applied at the end of 3 revolutions (1080 degrees). Then, in Figure 11 a correction of -5 degrees is applied 0.4 of a revolution later whereas, in Figure 12, the correction is not applied until one revolution later. This same case was run for correction delays of 0.1, 0.2, and 0.3 of a revolution with results which were nearly identical to figure 11. It can be seen that the quicker response in figure 11 does not appreciably reduce the peak flapping immediately after the input but the next peak is reduced by approximately 2 degrees in comparison to the slower response of Figure 12.

Effect of Feedback

Figures 8 through 12 typify the kinds of evaluations which were done prior to constructing the numerical controller previously discussed. After gaining confidence in the numerical flapping program and getting a feel for the feedback capability which would be needed, the final program was developed and parametric studies performed. To date, these have only been done for the AH-1J at 61 knots, 3075 ft. and 9500 lbs. ft. Some typical results illustrating the effects of the various parameters are shown in Figures 13 through 17. Table 1 lists the parameters used for each case noted on the figures. The parameters which are varied in these figures include:

a. The rate at which the controls can be moved.
b. The feedback increment to the controls each time excessive flapping is predicted.
c. The time required to predict the future flapping as a fraction of a rotor revolution.
d. The limit on the authority of the feedback controls.
e. The number of revolutions ahead through which the future flapping is predicted.

Figure 13 illustrates the effect of the rate of control movement. To put all rates and times in perspective, it is noted for the AH-1J that it takes 0.19 seconds per revolution or 5.2 revolutions per second. Thus, for example, at the rate shown in Figure 13 of 100 degs/sec, the controls will move 19.2 degrees in one revolution. From this figure, it would appear that the application rate of the controls is not a predominant effect. It is emphasized that this is the rate at which the controls are being moved to disturb the flapping and is not related to the rate of control feedback. The latter effect is accomplished in steps and is discussed later. It should be noted that the flapping limit for these calculations and for all of the results to follow was set at 8 degrees. Also, the disturbance for all of the cases was produced by applying 10 degrees of lateral cyclic at the end of two revolutions. In the case of Figure 13 it appears that the combination of looking ahead only two revolutions, feedback increments of 2 degrees, and an FPRED value of 0.2 results in a flapping which will exceed the flapping limit slightly. The effect of these parameters are discussed in the following paragraphs.

The feedback controls are applied as an accumulation of step inputs. A step is applied each time the controller predicts excessive flapping and is removed each time the controller predicts that the flapping will remain within limits. The effect of the feedback step size is illustrated in Figure 14. Here again, the influence of the step size is not a dominant parameter, at least for this particular operating state. However, it does appear as if a feedback increment of at least 4 degrees is needed to stay within the flapping limit.

The parameter, FPRED, is the fraction of a revolution required to predict the future flapping of the rotor over a given number of revolutions. Figure 15 illustrates the effect of varying this parameter over a range from 0.1 to 0.4 where the flapping is being predicted three revolutions ahead. The feedback step size for this figure is 4 degrees. The results shown here are somewhat puzzling and tend to contradict intuition. One would think that the lower FPRED the better. However, the graph shows a higher flapping for lower values of FPRED at around six revolutions. (although initially following the disturbance at around three revolutions the results are opposite) It may be that, what amounts to a high gain in the feedback, may be causing some type of an instability in the flapping.

Figure 16 shows the effect of limiting the total feedback control angle. From this figure it appears as if the authority of the feedback cannot be limited very much. For this particular case, as a result of the 10 degrees of lateral cyclic input, a total feedback of approximately 7 degrees must be allowed in order to stay within the prescribed flapping limit.

The effect of looking ahead 1, 2, and 3 revolutions in predicting the future flapping is shown in Figure 17. As one might expect, the differences in the results show up only in the first couple of peaks following the control input. Looking ahead only one revolution results in a flapping peak at about 2.8 revolutions which is approximately one degree higher than for the other two cases. It appears, from these and other cases, as if a value of NPRED of 2 or higher might be satisfactory.

References (Numerical Controller)

1. Hennis, R.P. and McCormick, B.W., A Computer Model for Determining Weapon Release Parameters for a Helicopter in Non-Accelerated Flight, U. S. Naval Surface Weapons Center, NSWC/DL TR-3823, October 1978.

2. Hennis, R.P. and McCormick, B.W., Computer Model for Predicting Dynamic Behavior of a Helicopter for Application to Weapons Delivery and Subsequent Safe Escape, U. S. Naval Surface Weapons Center, NSWC TR 85-285, September 1986.

3. McCormick, B.W., Aerodynamics of V/STOL Flight, Academic Press, New York, N. Y., 1967.

Nomenclature for the Numerical Controller Studies

The symbols given here are specific to the foregoing material since the study of the application of modern control theory was essentially independent of that for the numerical controller. A separate table of nomenclature giving the additional symbols for the analysis using modern control theory can be found at the end of that material.

Nomenclature for Text

SYMBOL	DEFINITION
α	Blade section angle of attack
α_s	Rotor disc plane angle of attack
β	Blade flapping angle measured from disc plane
Δt	Increment in time
ϵ	Distance from shaft axis to flapping hinge
ϕ	Angle of resultant velocity
θ	Blade section pitch angle
ψ	Blade azimuth angle
ω	Angular velocity of rotor
c	Blade section chord
C_d	Blade section drag coefficient
C_l	Blade section lift coefficient
D	Blade section drag
I_F	Blade moment of inertia about flapping axis
L	Blade section lift
M	Aerodynamic moment about flapping axis
M_W	Blade weight moment about flapping axis
r	radial distance from shaft axis
t	time
T	Blade thrust
V	Advance velocity

V_e	Resultant velocity at blade section
V_t	Tangential velocity at blade section
V_u	Upward velocity at blade section

Nomenclature for Programs

Variable Name	Definition
A1	longitudinal flapping
AI	temporary dummy variable in interpolation of airfoil table
ALPHA	disc plane angle of attack
ALPHAB	blade section angle of attack
ALPHAD (I)	section drag coefficient tabulated vs. this angle
ALPHAL (I)	section lift coefficient tabulated vs. this angle
AREA	disc area of rotor
B	number of blades
B1	lateral flapping
BETA	blade flapping angle relative to disc (shaft axis) plane
BETA0	coning angle
BETAD	first derivative with respect to time
BETADG	beta in degrees for printout
BETADI	initial value of BETAD when starting calculation
BETAI	initial value of BETA when starting prediction of flapping
BETLIM	limiting value which BETA is not to exceed
BI	temporary dummy variable in interpolation of airfoil table
C	blade section chord
C0	blade section root chord (X + 0)
CALPHA	cosine of alpha
CASE	identifying number
CBAR	mean chord of linearly tapered blade
CD	section drag coefficient
CDFAC	factor resolving drag in proper direction for extreme alpha
CDI (I)	tabulated section drag coefficient as function of ALPHAD (I)

CI	temporary dummy variable in interpolation of airfoil table
CL	section lift coefficient
CLFAC	factor resolving lift in proper direction for extreme alpha
CLI	tabulated section lift coefficient as function of ALPHAD (I)
CPSI	cosine of PSI
CRATE1	rate of increase of lateral cyclic, degs/sec
CRATE2	rate of increase of longitudinal cyclic, degs/sec
CT	rotor blade tip chord
D	rotor diameter
DDDR	derivative of section drag with respect to radius
DELBT1	amount by which BETA exceeds BETLIM
DELBT2	temporary vale of delbt1 in order to determine max value
DELFB1	increment to THETA1 when BETA is predicted to exceed limit
DELFB2	increment to THETA2 when beta is predicted to exceed limit
DELPSI	increment in azimuth angle for numerical integration
DELR	increment in radius for numerical integration
DELT	increment in time for numerical integration
DELT1	differential thrust as a function of radius for a given PSI
DELT2	same as, and averaged with, DELT1 to integrate for thrust
DELTH1	derivative of total blade thrust with azimuth angle, PSI
DELTH2	same as, and averaged with, DELTH1 to integrate for TAVG
DELTPR	increment in time for printout
DELX	increment in dimensionless radius for numerical integration
DI	temporary dummy variable in interpolation of airfoil table
DLDR	derivative of blade lift with radius
DMDR	radial derivative of blade aerodynamic moment at flap hinge
DTDR	radial derivative of blade thrust

DTR	factor to convert from degrees to radians
DWNWSH	subroutine to calculate average downwash
EPS	dimensionless distance of flapping hinge from rotor axis
EPSR	distance of flapping hinge from rotor axis
FB	switch to add or subtract feedback correction to theta
FPRED	fraction of revolution required to predict NPRED ahead
KBETA	rotor pitch-flap coupling (delta-3)
KNTRL	logic switch, see Figure 7
LAMDA	inflow ratio equals net flow up divided by tip speed
M1	radial derivative of aerodynamic moment about flapping hinge
M2	same as, and averaged with, M1 to integrate for moment
MAERO	moment about flapping axis produced by aerodynamic forces
MIF	blade mass moment of inertia about flapping axis
MU	ration of forward speed to tip speed, V/VT
MW	blade weight moment about flapping hinge
N	number of rotations at which control initiated
NMAX	maximum number of revolutions for run
NPRED	number of rotations to predict flapping ahead
OMEGA	rotational velocity of rotor, radians/sec
PERIOD	time for one rotor rotation
PI	the usual constant, 3.14159
PRT	number of calculations between printouts
PSI	azimuth angle
PSIDG	azimuth angle, degrees
PSII	initial value of PSI (see Figure 7)
PSIII	initial value of PSI (see Figure 7)
PSILIM	azimuth angle at which control initiated

PSIMAX	azimuth angle corresponding to NMAX
R	rotor radius
RHO	air mass density
SALPHA	sin of ALPHA
SIGMA	rotor section solidity equals B*C/PI/R
SPSI	sin of PSI
SUBCNTRL	subroutine provides a control input as a fvnction of TCON
SUBFLAP	subroutine integrates over R at PSI for flapping acceleration
TAU	parameter equals MW/MIF/OMEGA**2
TAVG	average rotor thrust in one revolution
TC	total elapsed time for predictin of future flapping
TCALC	integral of the rotor thrust with PSI for one revolution
TCON	elapsed time from when the control is first applied
TCONI	time when control is initiated
TH1DG	THETA1 in degrees for printout
TH2DG	THETA2 in degrees for printout
THE1I	initial value of THETA1
THE1LM	maximum incremental value for lateral control
THE2I	initial value of THETA2
THE2LM	maximum incremental value for longitudinal control
THETA	blade section pitch angle
THETA0	initial trim collective pitch
THETA1	initial trim lateral cyclic pitch
THETA2	INITIAL trim longitudinal cyclic pitch
THETAT	total blade twist
THRUST	instantaneous total blade thrust at a given value of PSI
TIME	elapsed time from start of run

TIMEI	time at which prediction started of future flapping
TOFF	length of time for control to be applied
TPRINT	time to print output
TWOPI	2*PI
V	rotor forward speed (advance velocity)
VR	resultant velocity at blade section from VU and VTHETA
VT	rotor tip speed due to OMEGA
VTHETA	velocity comonent at blade section in plane of rotation
VU	velocity at blade section normal to plane of rotation
W	average downwash corresponding to TAVG
WI	initial value of W at start of prediction of future flapping
X	dimensional radius
XH	value of X at hub

Design of a Digital Controller

As with the numerical approach, a dynamic simulation and the appropriate feedback calculations comprise the control algorithm. The dynamic simulation predicts the rotor state at a future time (usually three blade revolutions from the present), and the feedback controller determines if the flapping motion will be excessive; if so, the controller then automatically provides the necessary cyclic control step-input to limit the flapping. The helicopter dynamic model is the same as that used in the numerical control approach.

Implemented as a FORTRAN subroutine, the simulator serves the dual purpose of generating the actual rotor motion, and of predicting the future motion of the rotor, given the current rotor state and a control input. In the actual physical system, the rotor state and control input would, of course, be determined by appropriate sensors. It is envisioned that the complete controller (including the dynamic simulator) would be implemented in a small system of microprocessors.

This portion of the report describes the digital controller, presents the results of several simulations to test its performance, and concludes with a feasibility analysis of implementation using dedicated microprocessors.

Construction of the Closed-Loop Control System

The digital controller was designed using largely standard techniques, although the feedback gain matrix $\underline{K}$ was made time-varying in order to give better performance of the system. As with the numerical controller in the earlier part of this report, a simulator was used to generate the helicopter rotor flapping motion.

A block diagram of the complete system appears in Fig. 19. The difference between the simulator output and a reference signal, which is the physical constraint on safe flapping angle β, is taken as the feedback signal. The non-linear element (proportional gain with deadband) is used to obtain better stability properties of the system. A tracking filter is utilized to depress feedback signal noise and at the same time retain the ability to track varying inputs. A first-order hold is used for simulator input to achieve better tracking ability to varying inputs while zero-order holds are used elsewhere for simplicity, following standard design practice.

Design of Control Elements

1. Design of the first-order hold

The first-order hold is constructed with reference to Fig. 20. For a single input-single output (SISO) system, the hold is modeled as:

$$y(t) = y(t-1) + \frac{y(t-1) - y(t-2)}{t_{k-1} - t_{k-2}} \bullet (t-t_{k-1}), \quad t_{k-1} < t \leq t_k$$

where:

y = quantity being sampled

t = time

k = time index

This SISO hold is then extended to the multiple input-multiple output (MIMO) case.

2. Design of the Deadband

The deadband size is determined by system performance as well as tracking performance. The system performance here means the stability and transient characteristics of the system. From the stability point it is desirable to have a relatively large deadband size. Alternatively, from the perspective of tracking performance, the deadband size should be as small as possible, requiring a compromise.

As a practical consideration, because there is computing error in the simulator, it is undesirable to have the deadband size too small. In the actual implementation in a helicopter, sensor noise would result in the same consideration. For the given helicopter (AH-1J), the deadband Δ is chosen as 0.2 degree (as shown in Fig. 21). For simplicity, α and β are both chosen as 45° (i.e., the nonlinear element has unity gain); overall feedback gain is adjusted via the feedback gain matrix $\underline{K}$.

3. Design of the feedback gain matrix.

As is common to all control systems, the system stability is critically dependent on the value of feedback gain. Because the system performance is not easy to evaluate analytically and rotor response to a step input generally takes less than 3 revolutions to reach its steady state, it is sufficient to consider a first-order simulator for the present.

As expected, an improper choice of feedback gain can lead the system to diverge rapidly. To obtain minimum settling time of the system, the simulator has to have some overshoot to a step input. This requires that the loop gain cannot be too small. On the other hand, physical considerations do not allow overshoot in the system (e.g. an airframe strike could be the result); the feedback gain must not be so large that the rotor displays an oscillatory transient. To resolve this contradiction, a varying gain technique is used so that the gain decreases with time and reaches a steady state value. The initial value of the feedback gain can be chosen so that the simulator prevents an oscillatory transient. After several sampling points, the gain decreases so that the transient of the simulator is no longer oscillatory.

(1) Selection of initial value of feedback gain.

Considering the closed form solution of rotor flapping, it is helpful to understand the rotor behavior. From the closed form solutions, we have:

$$\beta = \beta_o - A_1 \cos \psi - B_1 \sin\psi$$

$$\frac{\partial\beta_0}{\partial\theta} = \frac{K_\beta f_4 \Gamma_F}{\Delta}, \qquad \frac{\partial\beta_o}{\partial\sigma_2} = \frac{\Gamma_F f_4}{\Delta}$$

$$\frac{\partial A_1}{\partial\theta_1} = \frac{K_\beta^2 \Gamma_F (A_{12} f_4 - A_{14} f_2) + A_{14} K_\beta}{\Delta}$$

$$\frac{\partial A_1}{\partial\theta_2} = \frac{1}{K_\beta} \frac{\partial A_1}{\partial\theta_1} = \frac{K_\beta \Gamma_F (A_{12} f_4 - A_{14} f_2) + A_{14}}{\Delta}$$

$$\frac{\partial A_1}{\partial\theta_1} = \frac{K_\beta f_2 \Gamma_F - 1}{\Delta}$$

$$\frac{\partial B_1}{\partial \Theta_2} = \frac{K_\beta{}^2\Gamma_F\ (A_{12}f_4 - A_{14}f_2) + A_{14}\ K_\beta + B_{11}\Gamma_F f_4}{\Delta}$$

where:

Θ_o = collective pitch angle

Θ_1 = lateral cyclic pitch angle

Θ_2 = longitudinal cyclic pitch angle

$$\Delta = (1 + K_\beta{}^2A_{14})\ (1 - K_\beta f_2\Gamma_F) + K_B f_4\Gamma_F\ (B_{11} + K_\beta{}^2A_{12})$$

A_1 = longitudinal flapping angle

B_1 = lateral flapping angle

β_o = coning angle

$K_\beta = \frac{\partial\theta}{\partial\beta}$

B = tip loss factor

$f_1 = \frac{B^3}{3}$

$f_2 = \frac{B^2}{4}(\mu^2 + B^2)$

$f_3 = B^3\ (\frac{B^2}{5} + \frac{\mu^2}{6})$

$f_4 = \frac{1}{3}\ \mu\ B^3$

$A_{11} = 4\ (\frac{\mu B^2}{2} - \frac{\mu^3}{8})$

$$A_{12} = \frac{8\mu B}{3(B^2 - \frac{\mu^2}{2})}$$

$$f_{13} = \frac{2\ \mu B^2}{B^2 - \frac{\mu^2}{2}}$$

$$A_{14} = \frac{B^2 + 3\frac{\mu^2}{2}}{B^2 - \frac{\mu^2}{2}}$$

$$\Gamma_F = \frac{C_o + C_T}{2} \cdot \rho \cdot 5.7\ (\tfrac{D}{2})^4 \cdot \tfrac{1}{2}\ /\ \text{MIF}$$

$$B_{11} = \frac{4\mu B}{3(B^2 - \frac{\mu^2}{2})}$$

MIF = blade moment of inertia about flapping axis

K_b = pitch-flap coupling coefficient.

For the example helicopter (AH-1J),

$$\frac{\partial\beta_o}{\partial\Theta_1} = 0.0\ ,\ \frac{\partial\beta}{\partial\Theta_2} = 0.14$$

$$\frac{\partial A_1}{\partial\Theta_1} = 0\ ,\ \frac{\partial A_1}{\partial\Theta_2} = 1.07$$

$$\frac{\partial\beta_1}{\partial\Theta_1} = -1\ ,\ \frac{\partial\beta_1}{\partial\Theta_2} = 0.035$$

Obviously,

$$\frac{\partial\beta}{\partial\Theta_1} \approx -\frac{\partial\beta_1}{\partial\Theta_1}\sin\psi \approx +\sin\psi$$

$$\frac{\partial\beta}{\partial\Theta_2} \approx -\frac{\partial A_1}{\partial\Theta_2}\cos\psi \approx -\cos\psi$$

Referring to Figs. (22a-22b), it is seen that the closed form expressions for

$$\frac{\partial\beta_o}{\partial\Theta_1}, \frac{\partial\beta_o}{\partial\Theta_2}, \frac{\partial A_1}{\partial\Theta_1}, \frac{\partial A_1}{\partial\Theta_1}, \frac{\partial\beta_1}{\partial\Theta_1}, \frac{\partial\beta_1}{\partial\Theta_2}$$

are close to the values calculated numerically. Similar agreement is obtained for other values of the parameters α, Θ_o, and ψ. A feedback gain of 1.0 gives a critically damped system; therefore, the initial value of feedback gain should be chosen larger than 1.0. The time-varying gain is computed as follows:

A Norm R defined for a vector (n x 1) is:

$$\text{Norm (U)} = U^T{}_{1/2} \begin{bmatrix} 1 & & & 0 \\ & e^{-c} & & \\ & & e^{-2c} & \\ 0 & & & e^{(n-1)c} \end{bmatrix} U_{1/2}$$

This measure puts some weighting on each element of U. Note that Norm (U) $\leq$ Euclidean norm of U.

Let U be a 10 x 1 vector matrix, and define

$$U(t,10) = E(t) \qquad U(t,9) = E(t-1)\,, \qquad U(t,1) = E(t-9) \qquad U(t) = \begin{bmatrix} |U(t,1)|^{1/2} \\ |U(t,2)|^{1/2} \\ \vdots \\ |U(t,10)|^{1/2} \end{bmatrix}$$

where E is the difference signal between the simulator output and reference signal, as shown in Fig. 19.

Then, the gain is deduced using:

$$\text{Gain}(t) = \text{Gain}(t-1)\,\frac{\text{Norm}\,[U(t)]}{\text{Norm}\,[U(t-1)]}$$

In order to keep the tracking ability, a lower limit is set on gain. In the case studied, c has been chosen as 0.1.

(2) Selection of lower limit on feedback gain.

The choice of lower limit on feedback gain is made on the basis that the simulator should not be oscillatory after some time t and the settling time of the simulator should be small. Obviously, a choice of dominant damping ratio of $\zeta = .707$ (i.e., $\sqrt{2}/2$) of the simulator dynamics is reinforced. In the given case, the lower limit is chosen as 0.6.

4. Design of the Tracking Filter

For the SISO system, the tracking filter is shown in Fig. 23(a), and the corresponding root locus appears in Fig. 23(b). This is a sampled-data system, and the design is performed accordingly using z-transforms. Two open-loop poles are placed at z = 1, so that the filer has zero steady state error in tracking a ramp input.

For simplicity, a fixed relation between a and k is chosen as

$$\frac{k}{k+1} = \frac{2a-2}{a-2}$$

where

k = tracking filter gain

a = open-loop zero

This relation has been determined to give good filtering performance.

The filter is then:

$$F(z) = \frac{2(a-1)\,z\,(z-a)}{(a-2)\,z^2 - 2a(a-2)\,z - a}$$

To illustrate the filtering effect of F(z), some random variations (noise) superimposed onto a deterministic ramp with a slope of 0.5 are generated and input to the system. Referring to Figs. (24a-24c), the tracking results are given for different choices of a.

It is seen that when a = 0.2, the filter response is fast enough to track the noise, whereas some noise depression effect is achieved when a = 0.8. As the filter response speed is further reduced, the filter will show less effect to noise and be more sluggish in tracking a varying true signal.

As a compromise, a = 0.7 is chosen. This SISO filter is then extended to our MIMO system.

Digital Simulation Results

Simulations were done with the sampling period being the time needed for the rotor blade to travel 30°.

It can be seen from Figs. (25-26) that the system can follow a step input reasonably with no overshoot across the physical flapping angle constraint line (the dot-dash line in the figures).

From Figs. (27-28), it is seen that there is a steady-state error to ramp inputs (i.e. the flapping actually reaches the physical limit). This steady-state error cannot be eliminated because the control is based on an imperfect prediction of future system response. Thus, no integral component in the loop can be used to eliminate the steady state error. Fortunately, this error is not very big and can be reduced if a higher-order hold is used for control inputs. Practically, a physical constraint on safe flapping angle can be set so that the ramp-tracking error of the closed-loop system may be allowed with no danger.

Implementation Requirements

Figure 29 is a block diagram of a typical microprocessor arrangement used in a control system. The input device includes sensors and an A/D converter, while the output device includes a D/A converter. The filter (digital to analog) is usually employed to reject noise from the senors and to smooth the control output. Coprocessors may be utilized to reduce the complexity of the software and/or increase the processing speed of the system through parallelism. For the helicopter blade flapping problem, the input-output variables are shown in Table 2.

1. Sampling Rates

For digital control, sampling is necessary and the choice of sampling rate is crucial to the performance of the system. The sampling of the measurement signals β and $\dot{\beta}$ must be based upon azimuthal angle ψ rather than being time-based, because in this application, β and $\dot{\beta}$ are periodic with respect to ψ. It was determined that a desirable sampling rate would be at least 5 times per revolution. Clearly, the value of the sampling interval $\Delta\psi$ must be less than 72 degrees. For a main-rotor tip-speed of approximately 740 ft/sec, and a blade diameter of approximately 44 ft, the sampling period (in time) must by less than 75 ms. Such a rate is readily achieved with existing technology.

2. Storage Requirements

There are approximately 150 variables to be stored in 24-bit RAM for the simulation. Information on C_L and C_D and all other helicopter parameters would be stored in approximately 100 words of 24-bit ROM. In addition, the trigonometric function "sin" would be stored in half-degree increments from 0 to 90 degrees in a look-up table (for 3-digit precision, this would require 180 words of 16-bit ROM). The controller requires another 50 variables in 24-bit RAM. The computational processor would need another 50 words of 24-bit RAM. Therefore, the total memory requirement for implementation is

RAM:	750 bytes
ROM:	660 bytes

which is a relatively trivial amount of memory for current technology. For example, the AMD 80C51 CMOS single-chip controller has 4K bytes of on-chip ROM and 128 bytes of on-chip RAM; therefore, only 1K bytes of external RAM are needed for this application.

3. Microprocessor Speed Requirement

To calculate the arithmetic operations done in one sampling period, the software flow-chart of the system to be implemented is shown in Fig. 29. The operations in each stage are arranged in Table 3. If parallel processing is employed for the inner loop in Fig. 29, the necessary operations are as shown in Table 4. To execute all of these instructions, an AMD 80C51 would require about 900 ms with its CPU operating at 12 MHz (working with 24-bit data).

Conclusions and Recommendations

It appears to be feasible to limit the amount of flapping of a helicopter rotor blade by means of a numerical controller which continuously predicts the future flapping of the rotor given the instantaneous state of the rotor and control input. Admittedly, this present study is limited in scope but enough has been learned about the response of the rotor to the proposed feedback system to justify proceeding further.

One of the major unanswered questions is that of the speed with which a microprocessor might do the calculations of the future rotor flapping. As an order of magnitude, it appears as if the microprocessor might be called upon to calculate the rotor flapping two revolutions ahead in the time it takes the rotor to move approximately 0.2 of a revolution, or around 40 milliseconds. This is certainly not out of the question, particularly considering the possibility that one might be able to use closed-form equations to predict the instantaneous aerodynamic moment on the blade. In this case, numerical integrations need only be done with respect to the azimuth angle.

For the digital controller, the time required for the microprocessor appears at first to be prohibitively large, but the implementation estimates were made using a relatively inexpensive "off the shelf" microprocessor. A custom-made chip could easily complete the required calculations in one-tenth to one-twentieth of the time estimated here. At the current rate of increase in processor speeds, it appears likely that a standard counterpart to the microprocessor analyzed here (AMD 80C51), with ten to twenty times its speed, will be available within a year or two. As with the numerical controller, further decreases in required computations might be achieved using closed-form equations for the blade-aerodynamics.

Unlike a stability augmentation system, the authority of the system presented here will probably have to be relatively high. However, this system will only provide a signal to cyclic control actuators at those rare times when the rotor is disturbed to such a degree as to result in excessive flapping.

If further work is pursued on this system, the following recommendations are made.

a. The equations of motion of the airframe should be included.

b. Considerably more cases should be examined, including trim at lower airspeeds with combinations of collective and cyclic control inputs.

c. The response of teetoring rotors should be compared with articulated rotors.

d. The time required for a microprocessor to do the required calculations should be studied realizing that, in the operating system, the controller will only have to predict the future flapping and not the continuous flapping as is done here for the simulation.

References (Digital Controller)

1. Wayne Johnson, Helicopter Theory, Princeton University Press, 1980.

2. Hugh F. Vanlandingham, Introduction to Digital Control Systems, MacMillan Publishing Company, 1985.

3. Goodwin, Graham Clifford, Adaptive Filtering Prediction, Control, Prentice-Hall, 1986.

4. Prouty, Raymond, Helicopter Performance, Stability, and Control. PWS Engineering, 1986.

Nomenclature for Digital Control Program (DCFLAP)

Note: Because the digital control program uses the same helicopter simulation code as the numerical control program, many of the variables are common to both. This table generally lists those variables that appear only in the digital program (DCFLAP). Refer to the Nomenclature for the Numerical Control Program for other variable definitions.

Variable Name	Definition
A11	temporary dummy variable
A12	temporary dummy variable
A13	temporary dummy variable
A14	temporary dummy variable
AIRFOIL	subroutine to calculate section lift and drag coefficient CL & CD
ALPHA	initial trim angle of attack of disk plane
ALPHAD	trim angle of attack of disk plane in degrees
ANORM	subroutine to calculate ANORM(U)
ANORM(U)	a special norm of a vector with data weighting
B11	temporary dummy variable
BEEP	logical variable, signaling occurrence of excessive flapping
BET	value of BETA in degrees for printout or writing to initial condition table
BET10	initial trim flapping angle in radians at PSIBD
BET10D	temporary dummy variable
BET1D1	temporary dummy variable in interpolation of initial condition table
BET1D2	temporary dummy variable in interpolation of initial condition table
BET20	first derivative of flapping angle corresponding to BET10
BET20D	temporary dummy variable
BET2D1	temporary dummy variable in interpolation of initial condition table
BET2D2	temporary dummy variable in interpolation of initial condition table
BETD	value of BETAD in degs/sec for printout or writing to initial condition table

BWFLAP	subroutine to simulate the rotor blade flapping motion
CDELAY	temporary dummy variable
CNTRL	subroutine formulating control law
CNTRL1	feedback signal of lateral cyclic pitch in degrees for printout
CNTRL2	feedback signal of longitudinal cyclic pitch in degrees for printout
CRATEA	rate of increase of angle of attack of disk plane
CRATEO	rate of increase of collective pitch
D0	perturbation in collective pitch
D1	perturbation in lateral cyclic pitch
D2	perturbation in logitudinal cyclic pitch
DALPHA	perturbation in angle of attack of disk plane
DAMP	damping ratio of rotor blade flapping
DELAY	phase angle delay of rotor blade flapping
DENOM	temporary dummy variable
DETECT	logic switch, initiating perturbation if NREV reaches PERTB
E1	THETA1 in degrees for printout
E2	THETA2 in degrees for printout
F1	temporary dummy variable
F2	temporary dummy variable
F3	temporary dummy variable
F4	temporary dummy variable
FDBACK	subroutine to determine the amount of feedback
GAIN	magnitude of feedback gain vector, see Fig. 19 (the system block diagram)
IKTEN	logic switch for sampling
KTEN	sampling is done once every KTEN integration points
NBET	number of points per revolution in initial condition table
NENTRY	entry point in initial condition table

NREV	azimuth angle PSI in terms of number of revolutions from the start of simulation
OVER	amount of excessive flapping
OVER0	maximum amount of excessive flapping
PSI00	temporary dummy variable
PSI1ST	temporary dummy variable in preparing the initial condition table
PSIBD	azimuth angle in degrees when simulation begins
PSIBD1	temporary dummy variable in interpolation of initial condition table
PSID	value of PSI in degrees for printout or writing to initial condition table
REPLY	logic switch, user keyboard input
SOS	azimuth position at which maximum excessive flapping occurs
STEP	step size of PSI in degrees in the initial condition table
T1	temporary dummy variable
T2	temporary dummy variable
T3	temporary dummy variable
T4	temporary dummy variable
THET0	initial trim collective pitch
THET0D	trim collective pitch in degrees
THET1	initial trim lateral pitch
THET1D	trim lateral cyclic pitch in degrees
THET2	initial trim longitudinal pitch
THET2D	trim longitudinal cyclic pitch in degrees
THRST1	temporary dummy variable
THRST?	temporary dummy variable
TRIM	logical variable,specifying the status of existence of initial condition table
TTAVG	temporary dummy variable
U0(I)	similar to U(I) but used as a temporary one

UTHET1(I)	temporary dummy array, used in forming first-order hold
UTHET2(I)	temporary dummy array, used in forming first-order hold
U(I)	vector windowing the flapping angle
VAR1	temporary dummy variable
VAR2	temporary dummy variable
VAR3	temporary dummy variable
VNUL	temporary dummy variable
WNREL	relative natural frequency of rotor blade flapping w.r.t OMEGA
XA1	temporary dummy variable
XB1	temporary dummy variable

TABLE 1

Parameters Corresponding to Figures 13 through 17

CASE	DELFB1	DELFB2	FB1LIM	FB2LIM	NPRED	FPRED
1*	2	2	5	5	1	.2
2*	2	2	5	5	1	.2
3	2	2	5	5	1	.2
4	2	2	5	5	2	.2
5	2	2	5	5	3	.2
6	4	4	8	8	3	.1
7	4	4	8	8	3	.2
8	4	4	8	8	3	.4
9	4	4	8	8	2	.2
10	3	3	8	8	2	.2
11	2	2	8	8	2	.2
12	2	2	6	6	2	.2
13	2	2	7	7	2	.2
14	4	4	8	8	3	.1
15**	2	2	8	8	2	.2
16	4	4	7	7	1	.2
17	4	4	7	7	2	.2
18	4	4	7	7	3	.2

FOR ALL CASES

BETLIM	= 8	(*BETLIM = 50)
TOFF	= 10	
PRT	= 3	
NMAX	= 10	
CRATE1	= 100	(**CRATE1 = 200)
CRATE2	= 100	(**CRATE2 = 200)

TABLE 2

Input-Output Variables for Digital Controller

VARIABLE	INPUT	OUTPUT
Downwash, w	X	
Thrust, T	X	
Azimuth position, ψ	X	
Flapping angle, β	X	
Flapping rate, $\dot{\beta}$	X	
Forward speed, v	X	
Angle of attack, α	X	
Collective, θ_o	X	
Lateral cyclic, θ_1	X	X
Longitudinal cyclic, θ_2	X	X
Air density, ρ	X	

TABLE 3

Operations in Each Stage

STAGE	ADD	SUB	MUL	DIV	REF.
1	4	6	29	12	9
2	2	3	11	5	9
3	15	15	35	12	17
4	69	55	105	22	34

STAGE	CMP	BRANCH	ABS	SORT	OPS
1	0	0	0	1	0
2	0	0	0	0	0
3	111	3	4	1	4
4	10	7	5	0	3

TABLE 4

Operations in One Sample Period

ADD	SUB	MUL	DIV	MEM. REF.
90,000	7,800	18,000	5,100	6,900

CMP	BRANCH	ABS	SORT	BINARY OPS
12,100	1,000	900	200	700

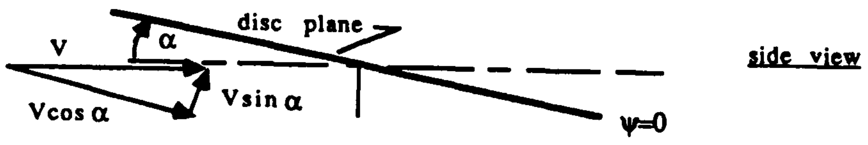


FIGURE 1
LEFT SIDE VIEW OF A ROTOR SHOWING THE
ANGLE OF ATTACK AND FREESTREAM VELOCITY COMPONENTS
PARALLEL AND NORMAL TO THE DISC PLANE

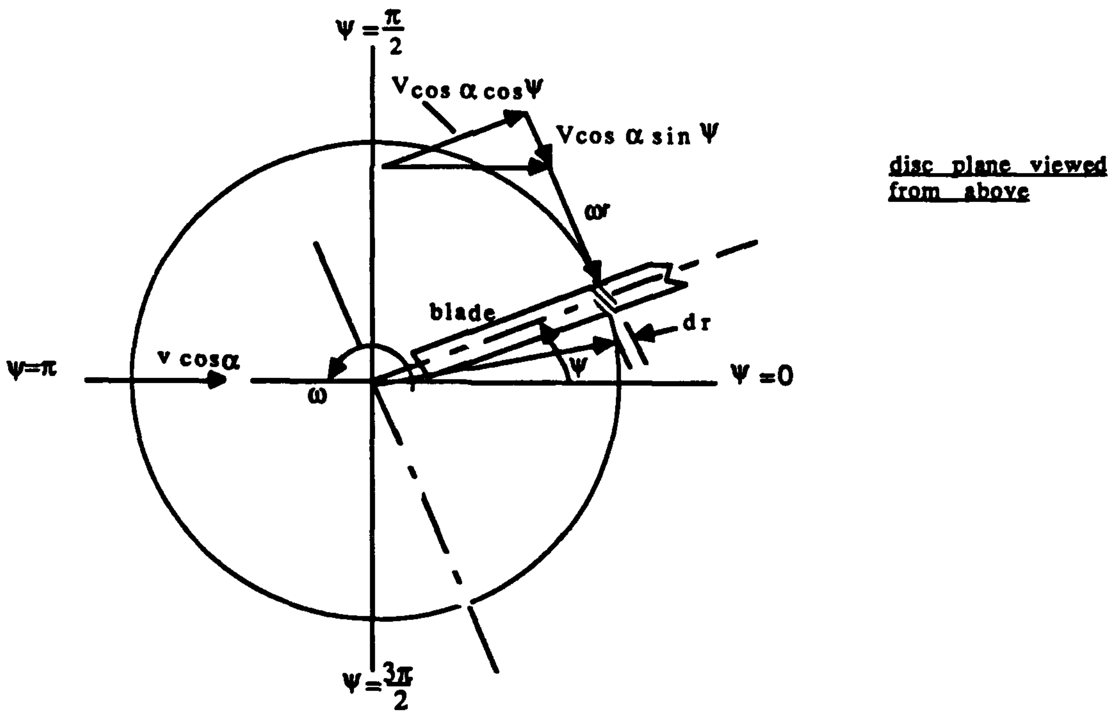
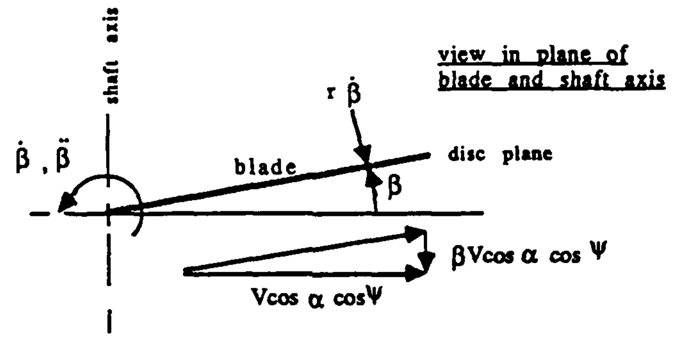
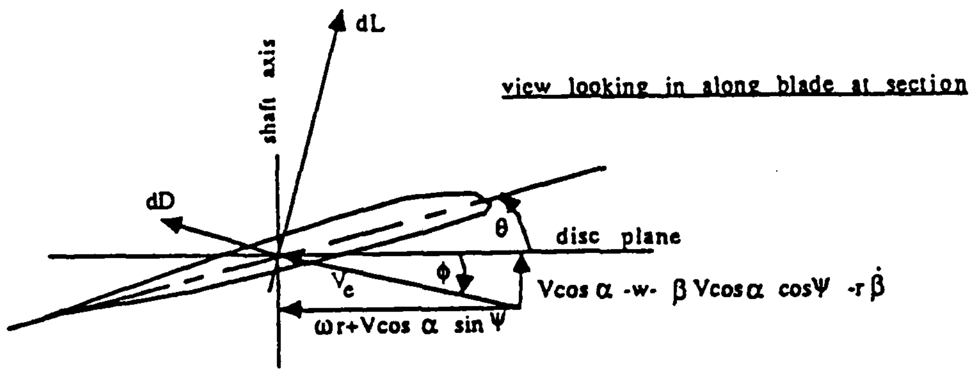


FIGURE 2
TOP VIEW OF THE DISC PLANE SHOWING VELOCITY
COMPONENTS PARALLEL AND NORMAL TO THE BLADE



shaft axis
view in plane of blade and shaft axis
$r\dot{\beta}$
$\dot{\beta}, \ddot{\beta}$
blade
disc plane
β
$\beta V\cos\alpha \cos\Psi$
$V\cos\alpha \cos\Psi$

FIGURE 3
BLADE - SHAFT AXIS PLANE SHOWING VELOCITY COMPONENTS RESULTING FROM FLAPPING AND DOWNWASH



dL
shaft axis
view looking in along blade at section
dD
θ
disc plane
V_e
ϕ
$V\cos\alpha$ -w- $\beta V\cos\alpha \cos\Psi$ $-r\dot{\beta}$
$\omega r+V\cos\alpha \sin\Psi$

FIGURE 4
BLADE AIRFOIL SECTION SHOWING THE TANGENTIAL AND NORMAL VELOCITIES TO THE DISC PLANE WHICH DETERMINE THE SECTION ANGLE OF ATTACK

FIGURE 5
RESOLUTION OF SECTION LIFT AND DRAG
FOR EXTREME ANGLES OF ATTACK

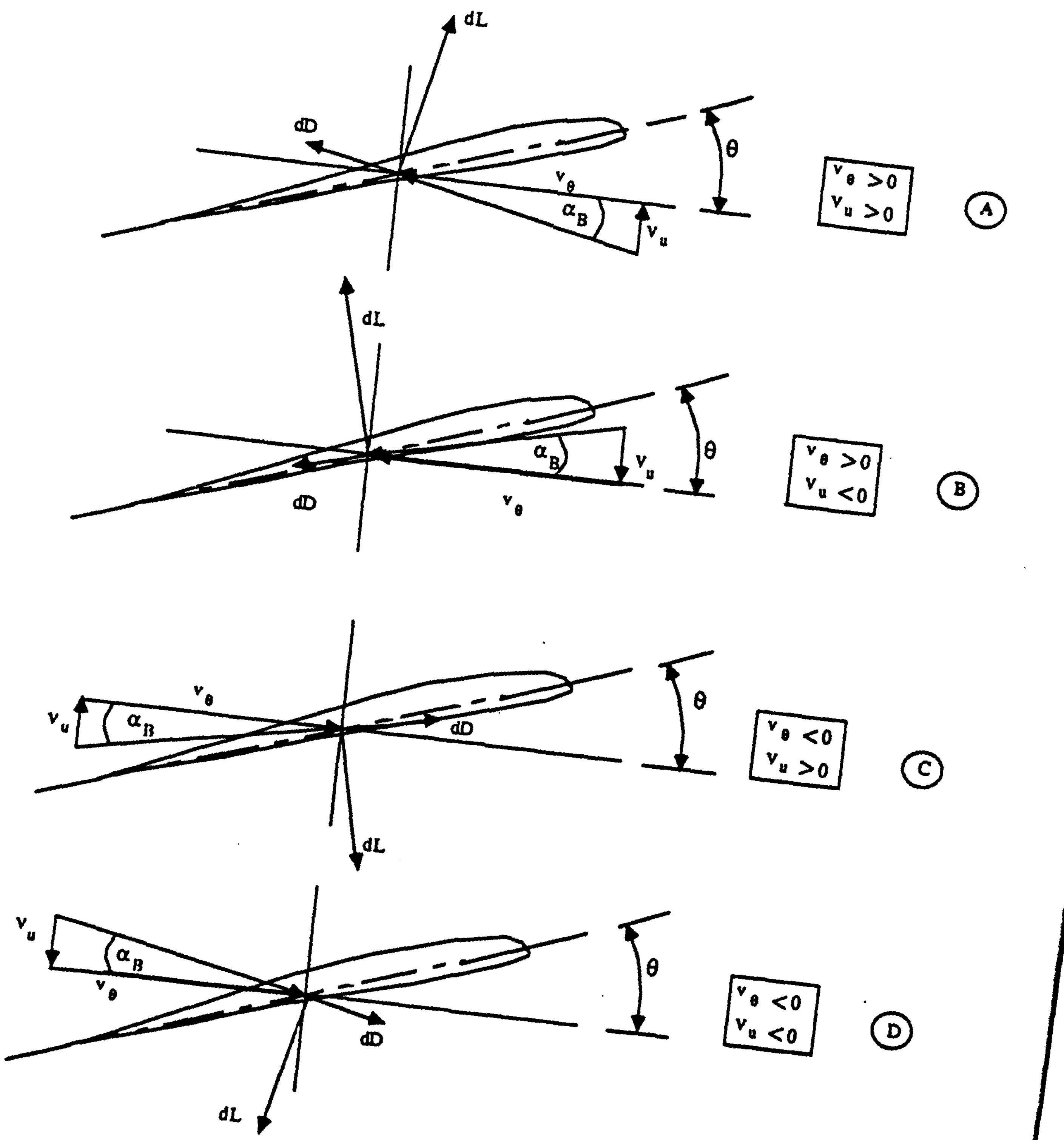


FIGURE 6
SCHEMATIC OF THE CONTROLLER ACTION

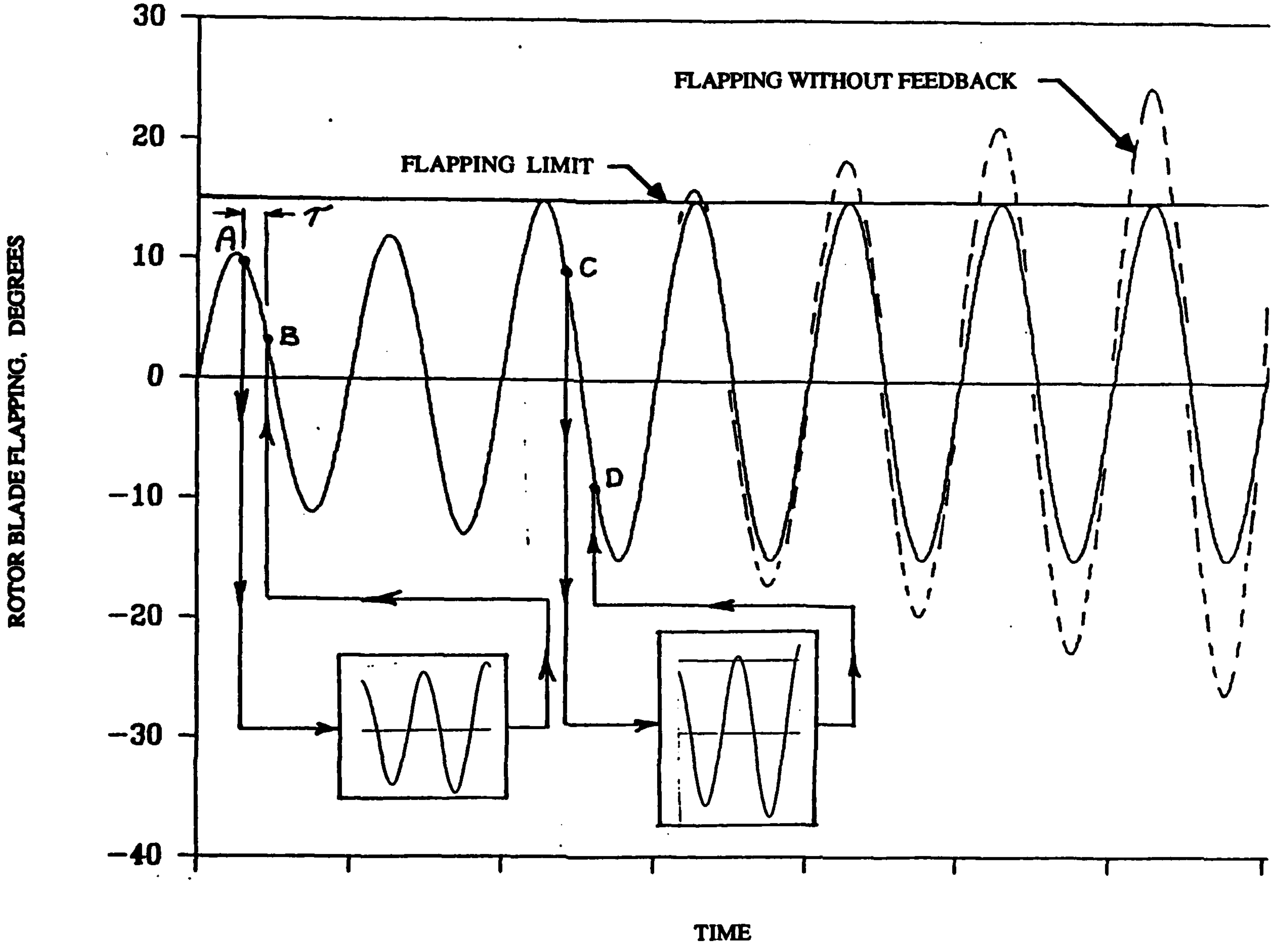


FIGURE 7
LOGIC FOR THE PROGRAM WHICH SIMULATES ROTOR FLAPPING AND CONTROLLER ACTION

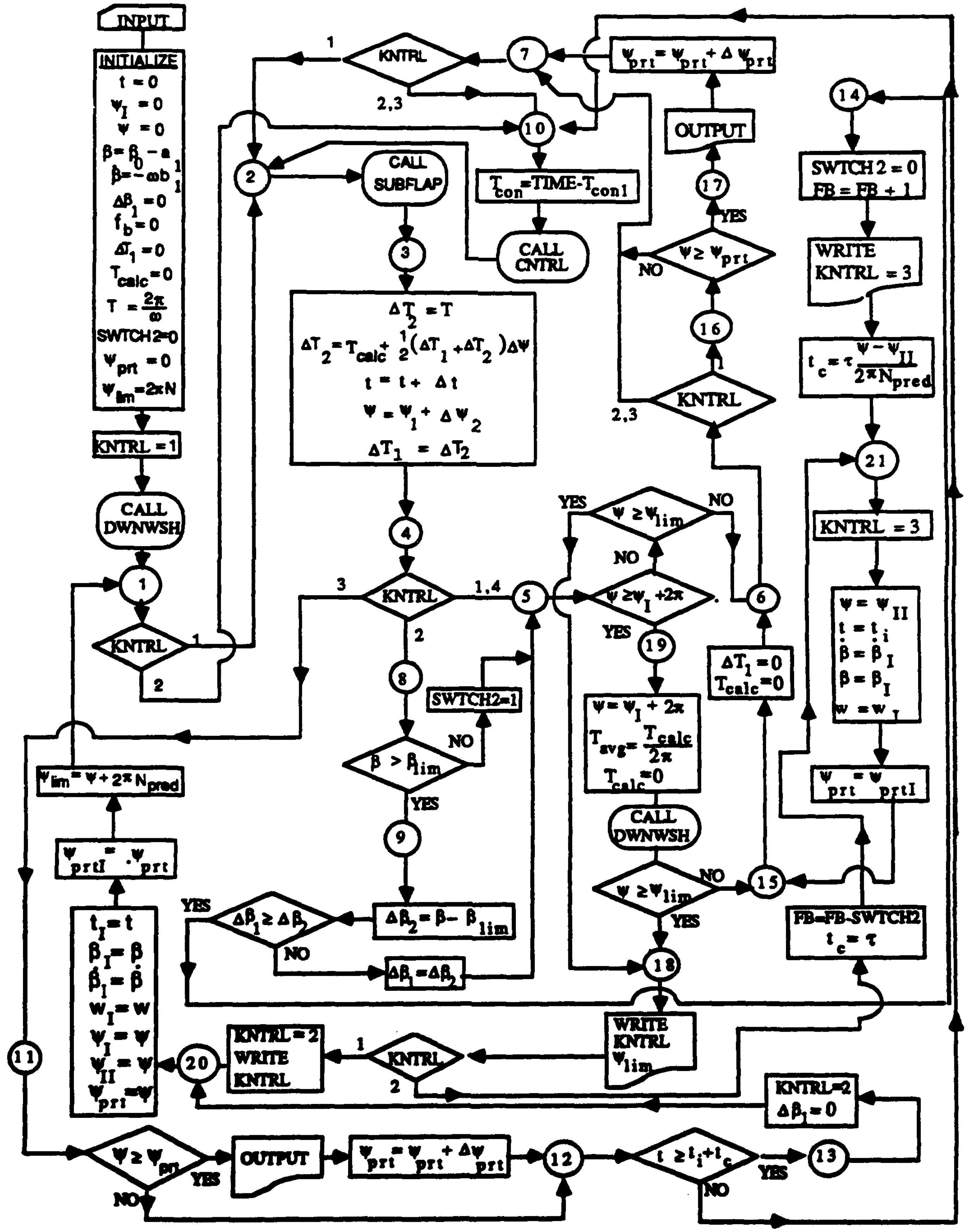


FIGURE 8
ROTOR BLADE FLAPPING PREDICTED BY THE NUMERICAL PROGRAM COMPARED WITH PREDICTIONS BASED ON CLOSED - FORM EQUATIONS
(AH-1J trimmed at 61 kts., 9500 lbs. and 3075 ft.)

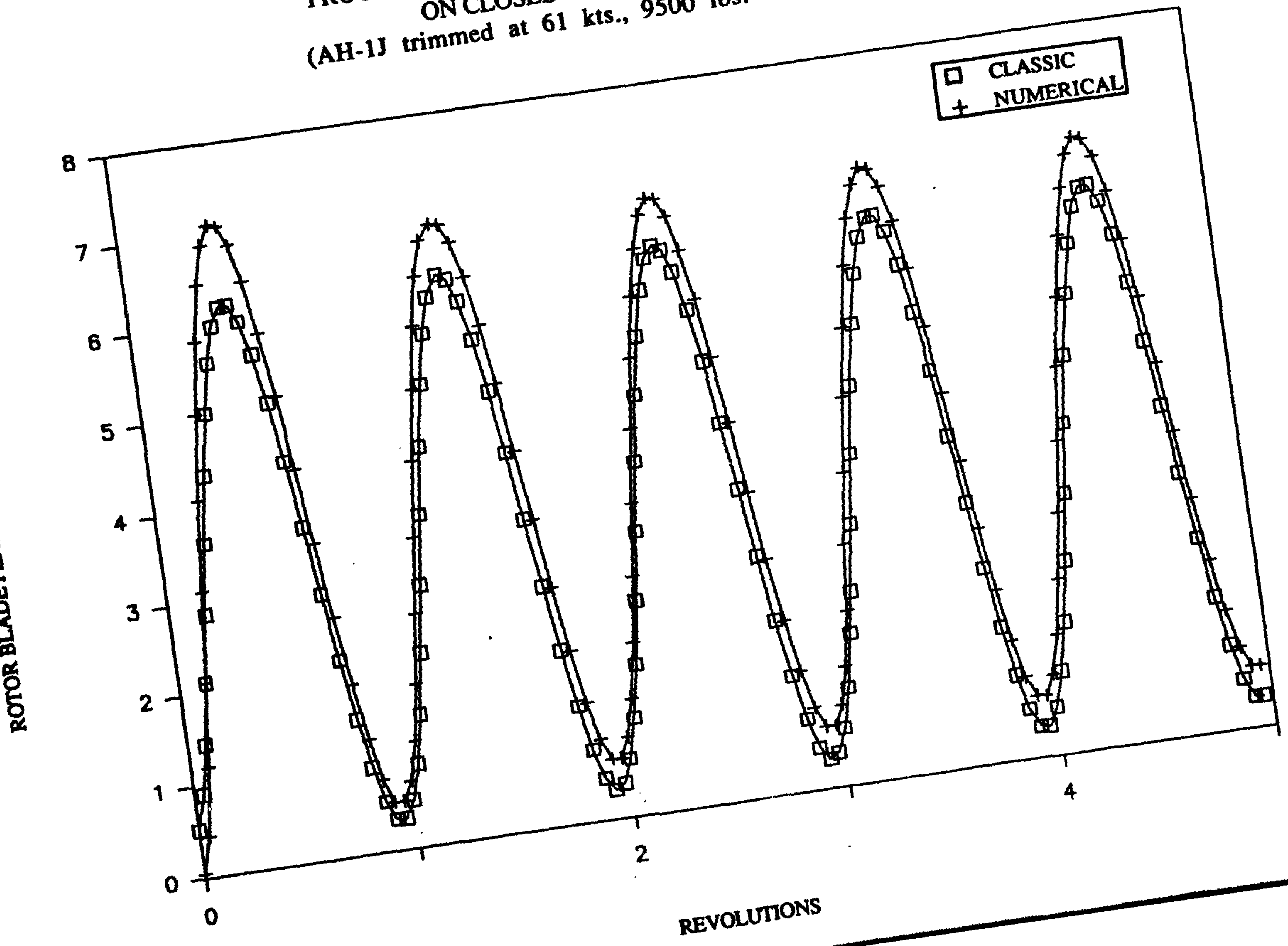


FIGURE 9
ROTOR BLADE FLAPPING COMPARISON FOR THE SAME CONDITIONS AS FIGURE 8 EXCEPT THAT TEN DEGREES OF LATERAL CYCLIC IS APPLIED AFTER TWO REVOLUTIONS

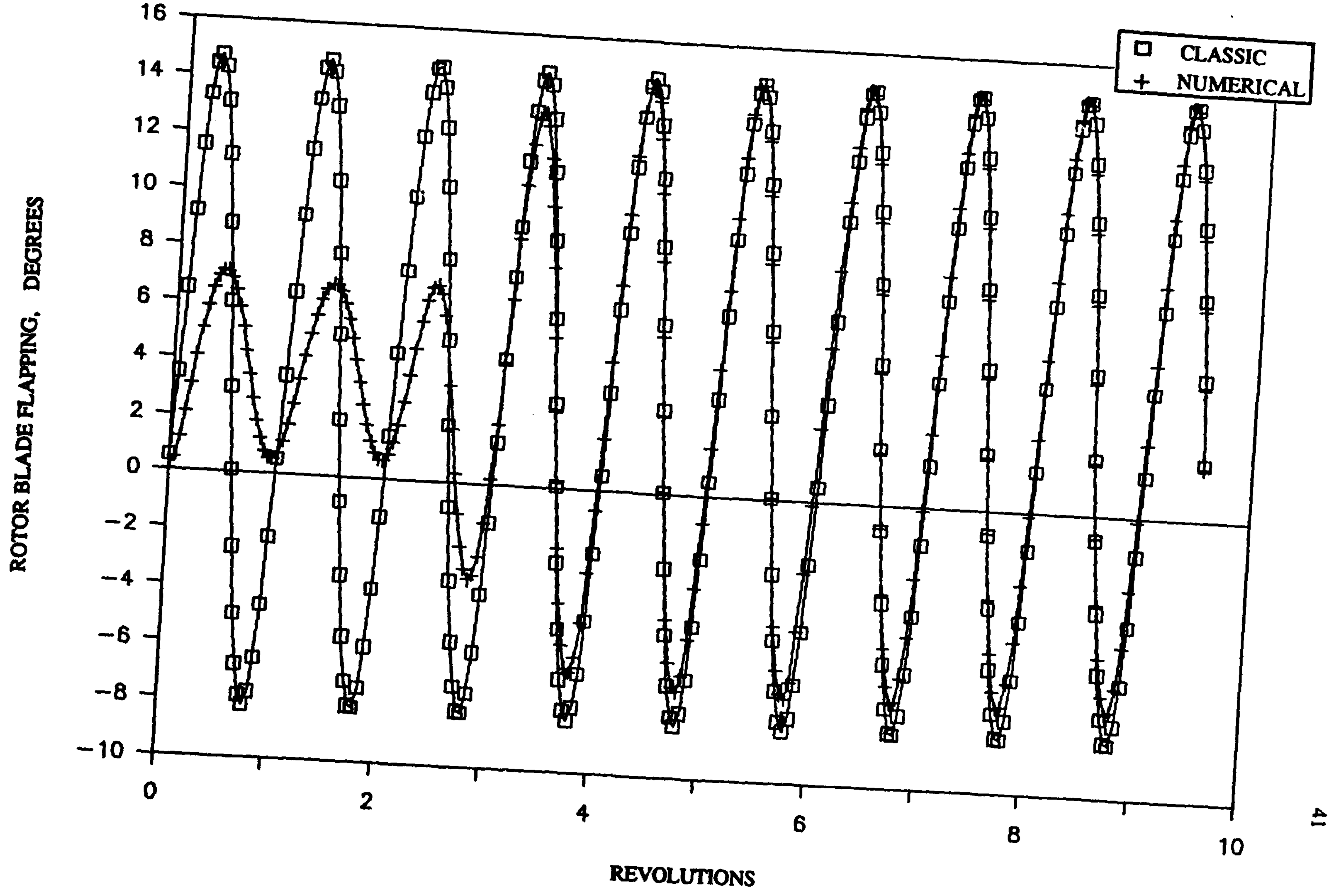


FIGURE 10
PREDICTION OF ROTOR BLADE FLAPPING RESPONSE
TO A LATERAL CYCLIC CONTROL INPUT OF 5 DEGREES
AT 3 AND 3.5 REVOLUTIONS
(AH-1J trimmed at 80 kts., sea level and 9500 lbs.)

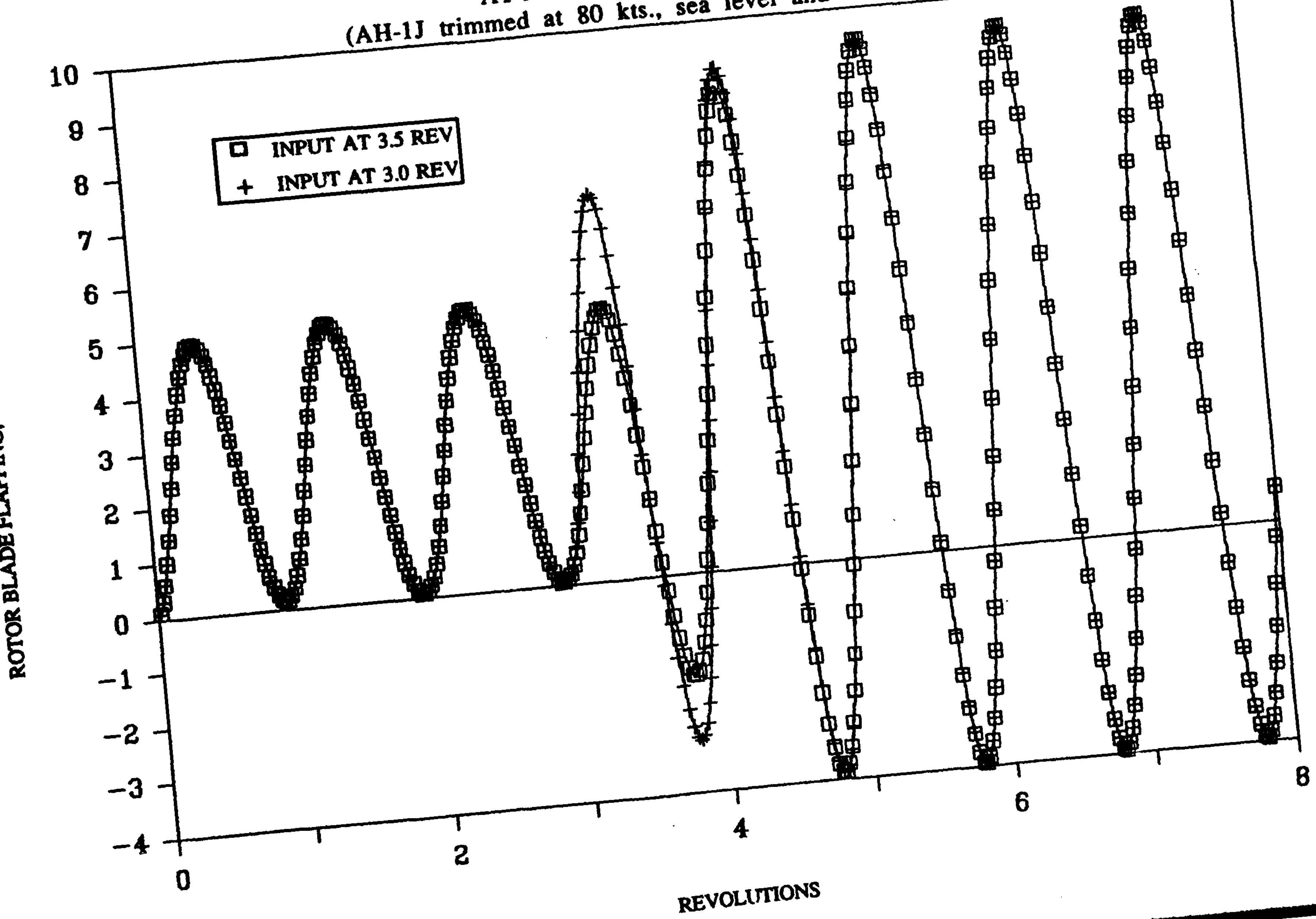


FIGURE 11
PREDICTION OF ROTOR BLADE FLAPPING RESPONSE TO A LATERAL CYCLIC INPUT OF 5 DEGREES APPLIED AT N=3 AND REMOVED FOUR-TENTHS OF A REVOLUTION LATER
(AH-1J trimmed at 80 kts., sea level and 9500 lbs.)

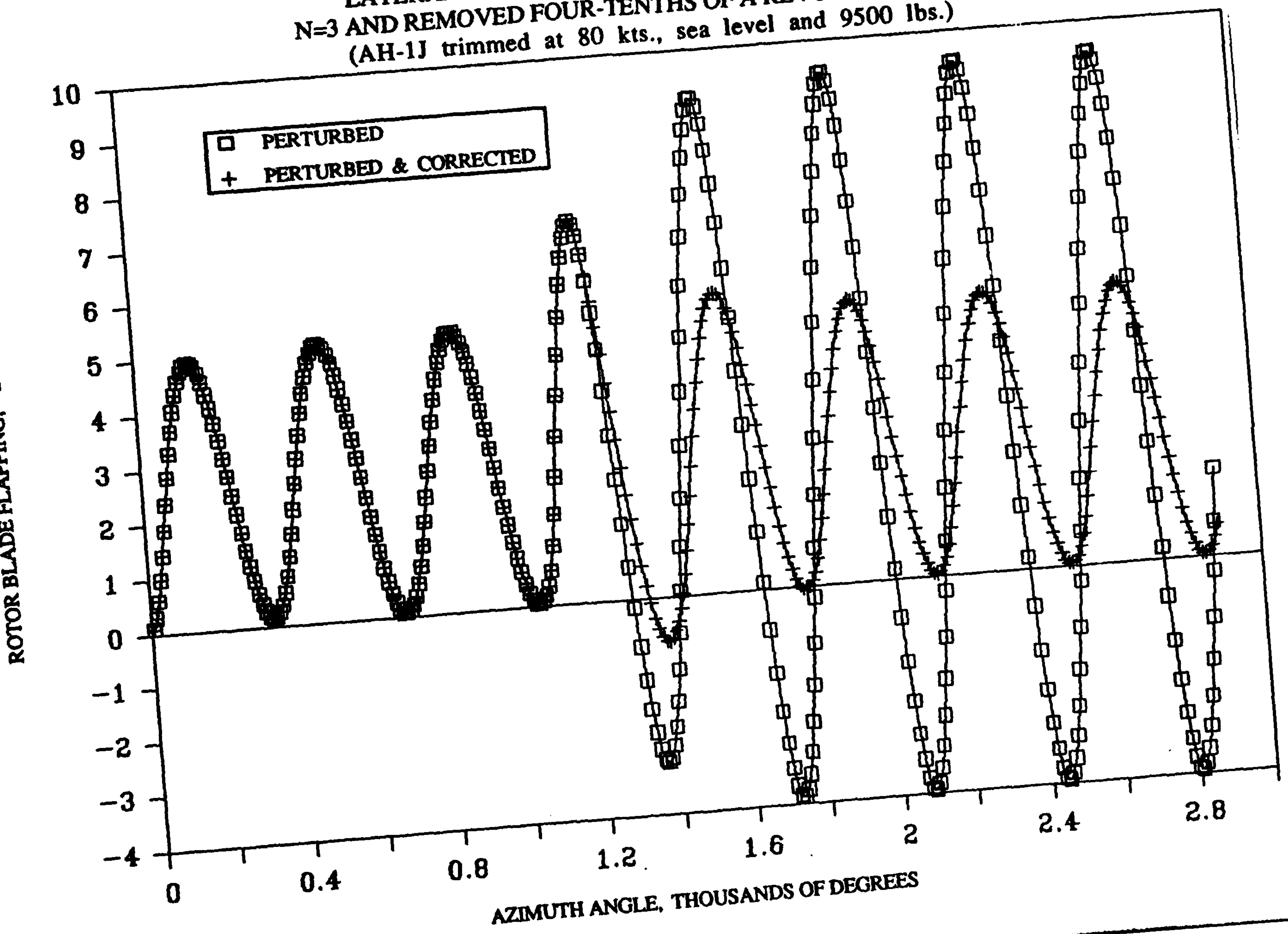


FIGURE 12
PREDICTION OF ROTOR BLADE FLAPPING RESPONSE TO A LATERAL CYCLIC INPUT OF 5 DEGREES APPLIED AT N=3 AND REMOVED ONE REVOLUTION LATER
(AH-1J trimmed at 80 kts., sea level and 9500 lbs.)

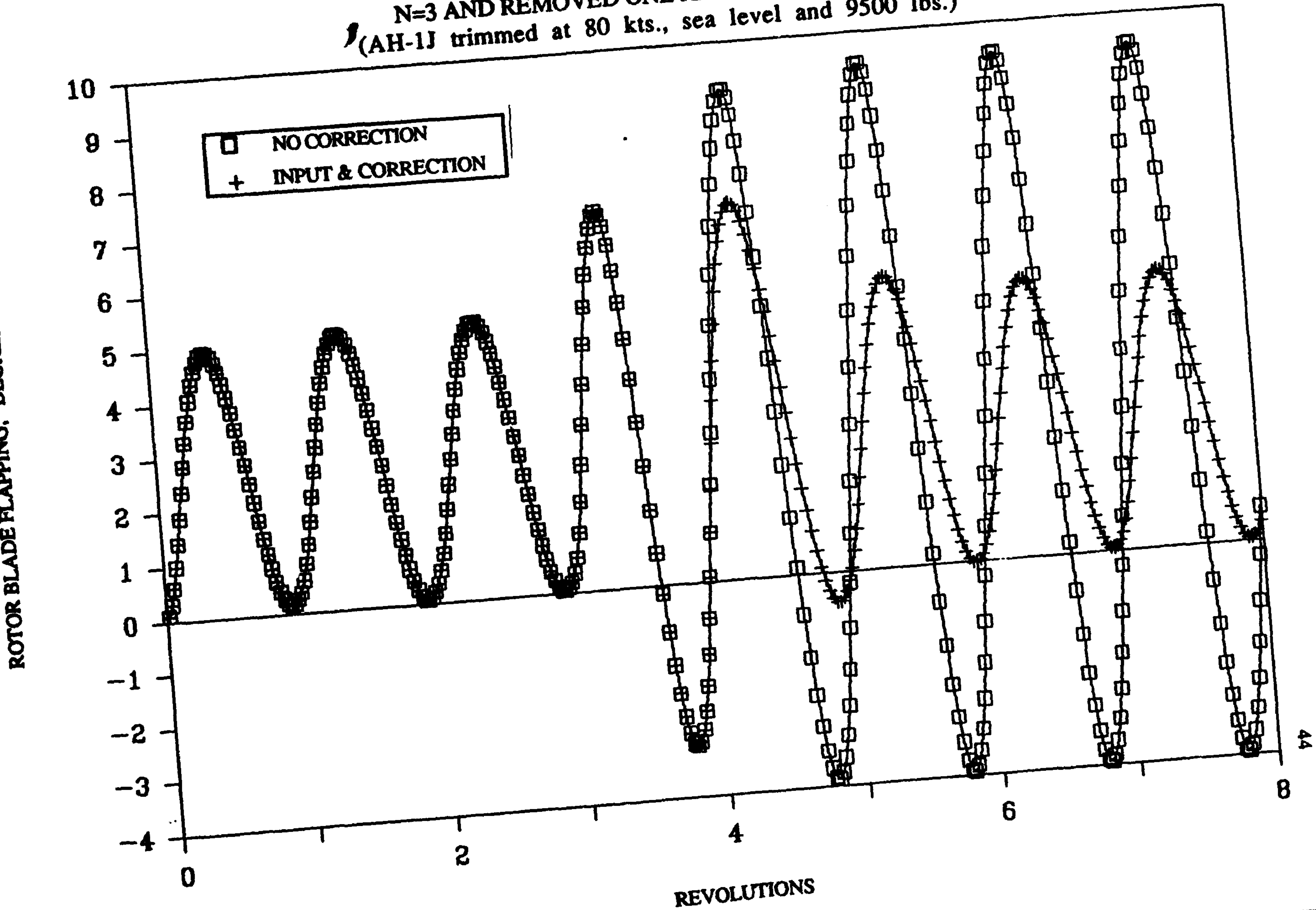


FIGURE 13
EFFECT OF DISTURBING ROTOR BLADE FLAPPING BY CONTROL INPUT AT DIFFERENT RATES
(Cases 15 and 16)

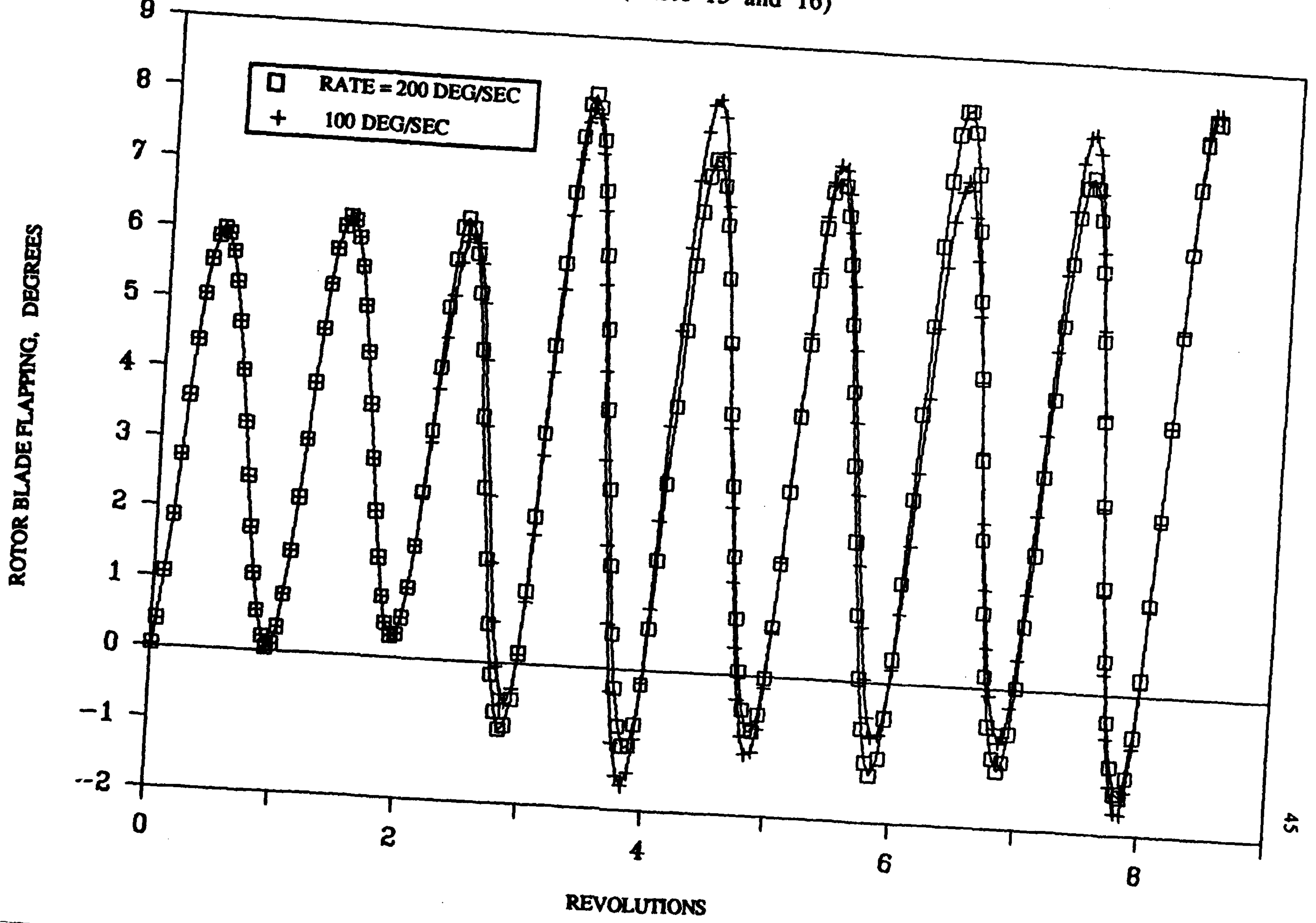


FIGURE 14
EFFECT OF FEEDBACK STEP SIZE ON CONTROLLER PERFORMANCE
(Cases 9, 10 and 11)

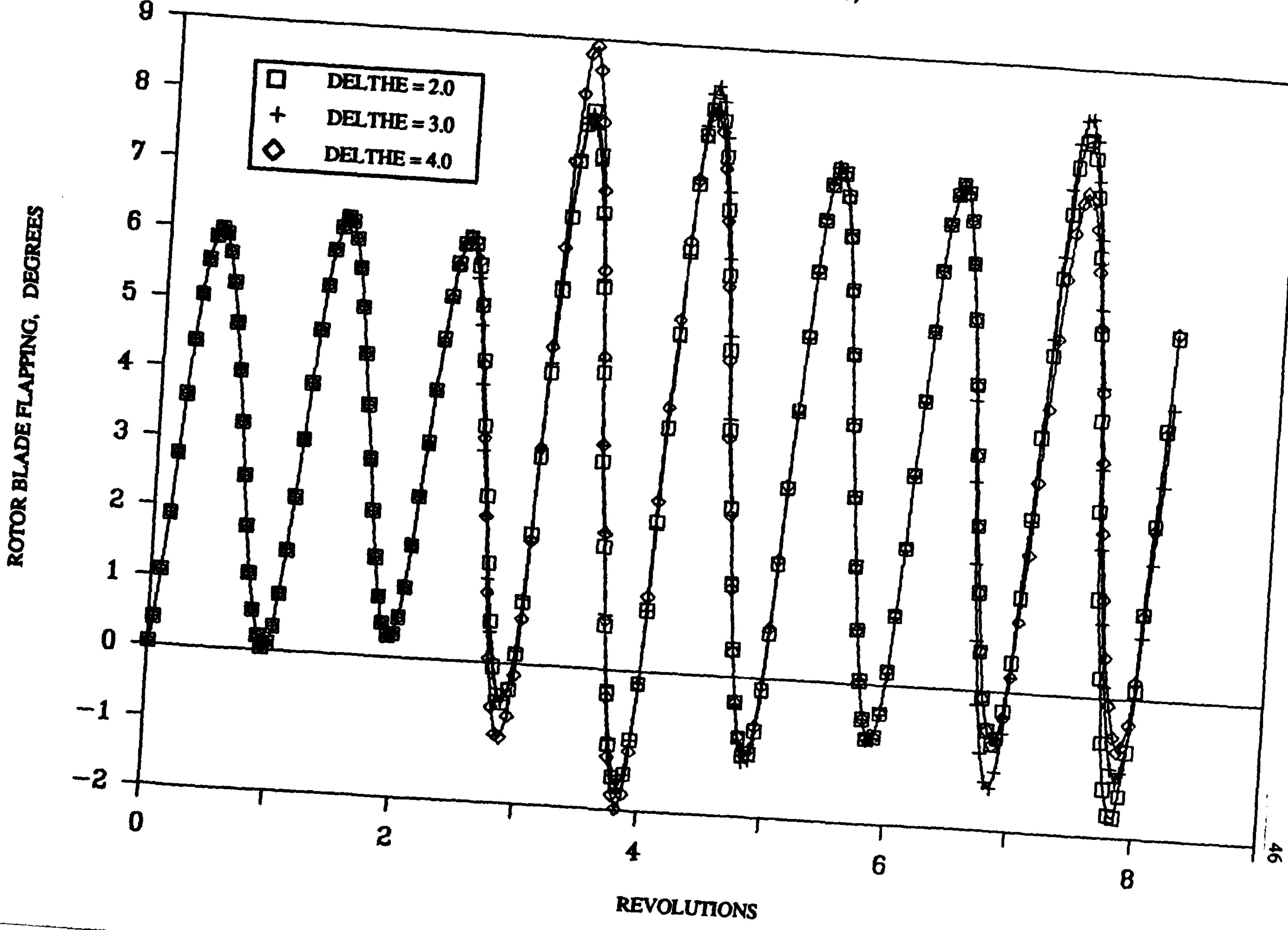


FIGURE 15
EFFECT OF ASSUMING DIFFERENT TIMES FOR CALCULATING
FUTURE FLAPPING OVER A GIVEN NUMBER OF REVOLUTIONS
(Cases 6, 7 and 8)

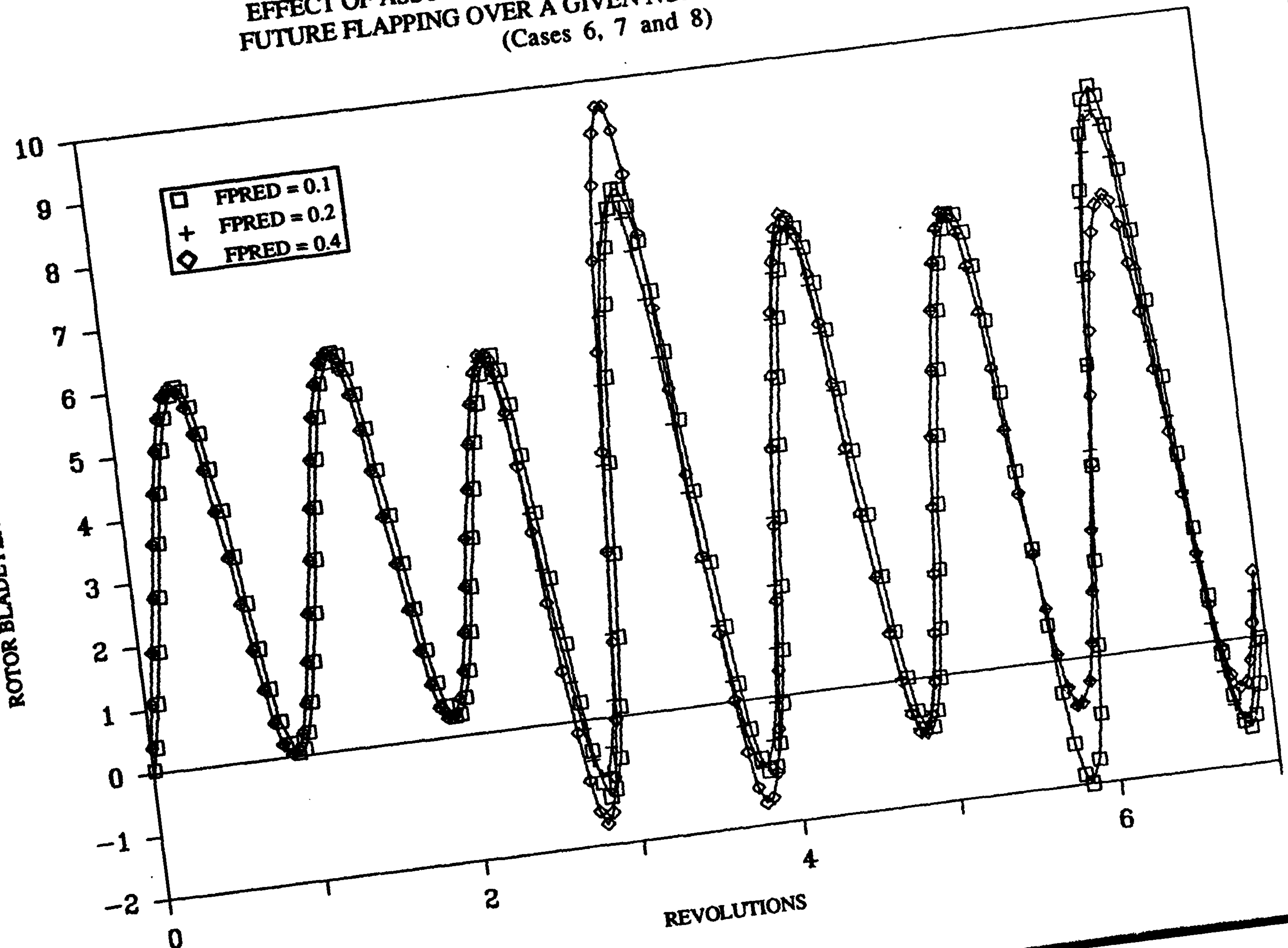


FIGURE 16
EFFECT OF LIMITING THE AMPLITUDE OF THE FEEDBACK CONTROL ON CONTROLLER PERFORMANCE
(Cases 4, 12, 13 and 15)

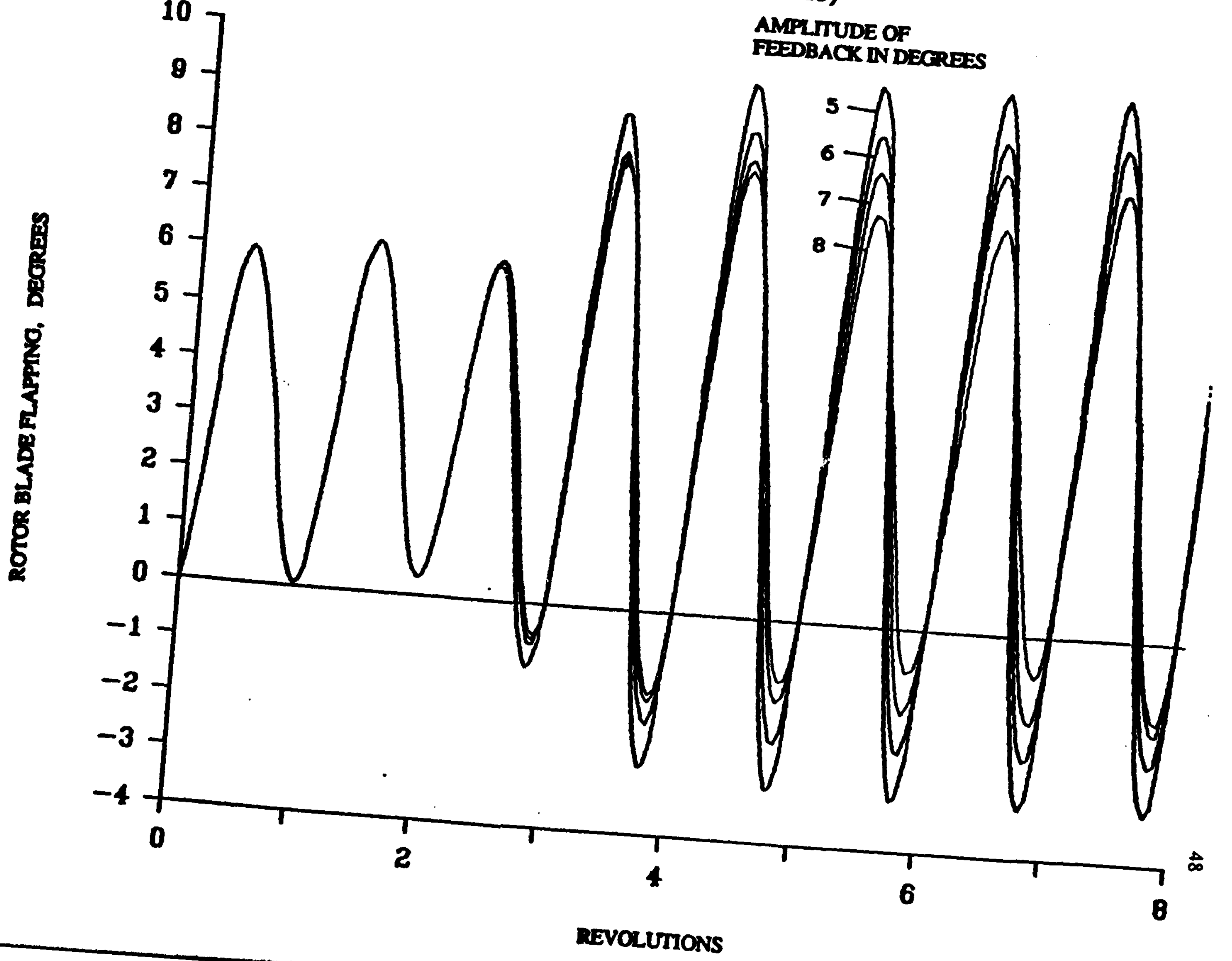
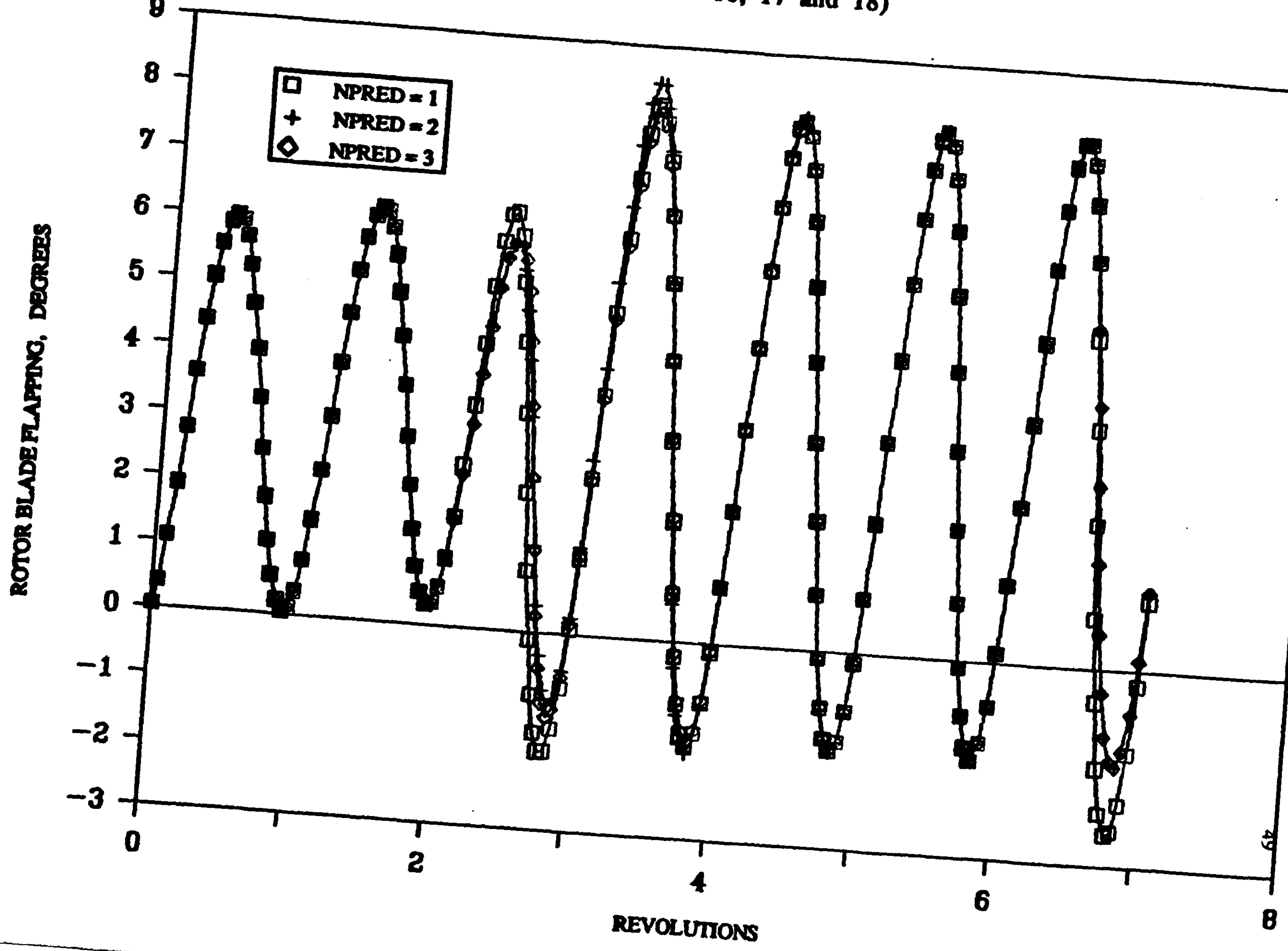


FIGURE 17
EFFECT OF THE NUMBER OF REVOLUTIONS AHEAD FOR WHICH THE FUTURE FLAPPING IS PREDICTED
(Cases 16, 17 and 18)



NOTE: FIGURE 18 INTENTIONALLY OMITTED.

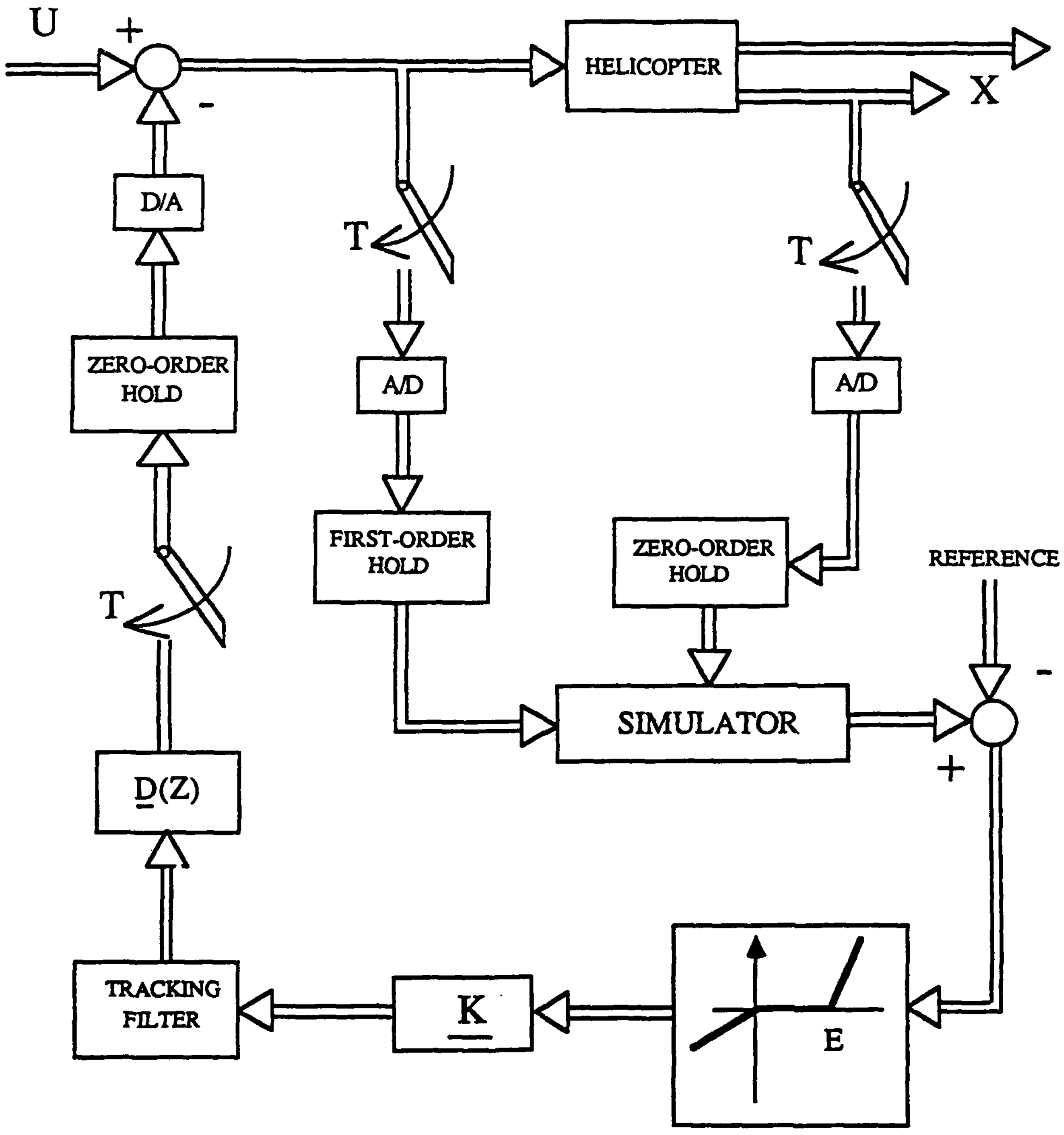


Figure 19 Block diagram of the digital control system.

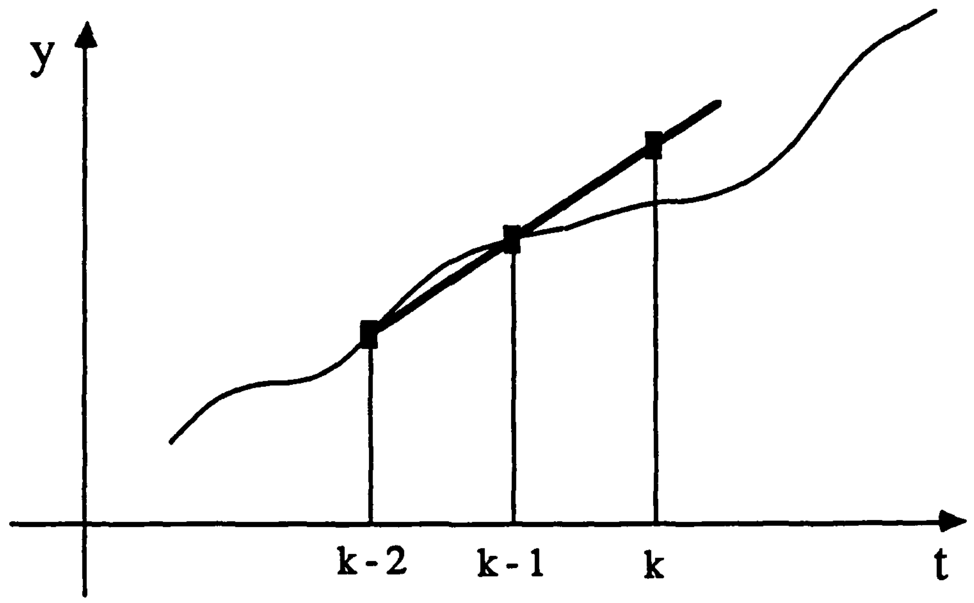


Figure 20 First-order hold model

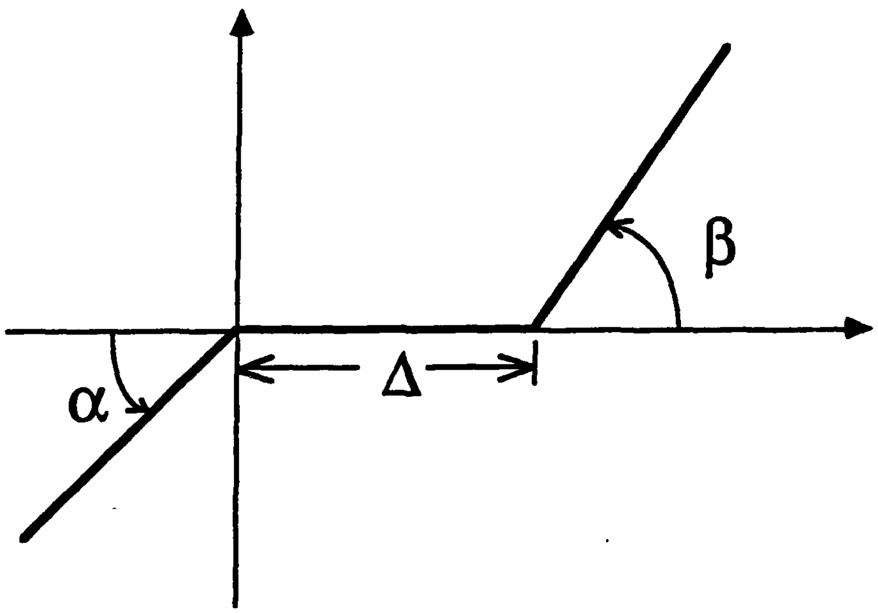


Figure 21 Nonlinear element

Figure 22a Steady-state rotor flapping angle β
$\alpha = -5$ deg. $\theta_0 = 15$ deg. $\Psi = 0$ deg.

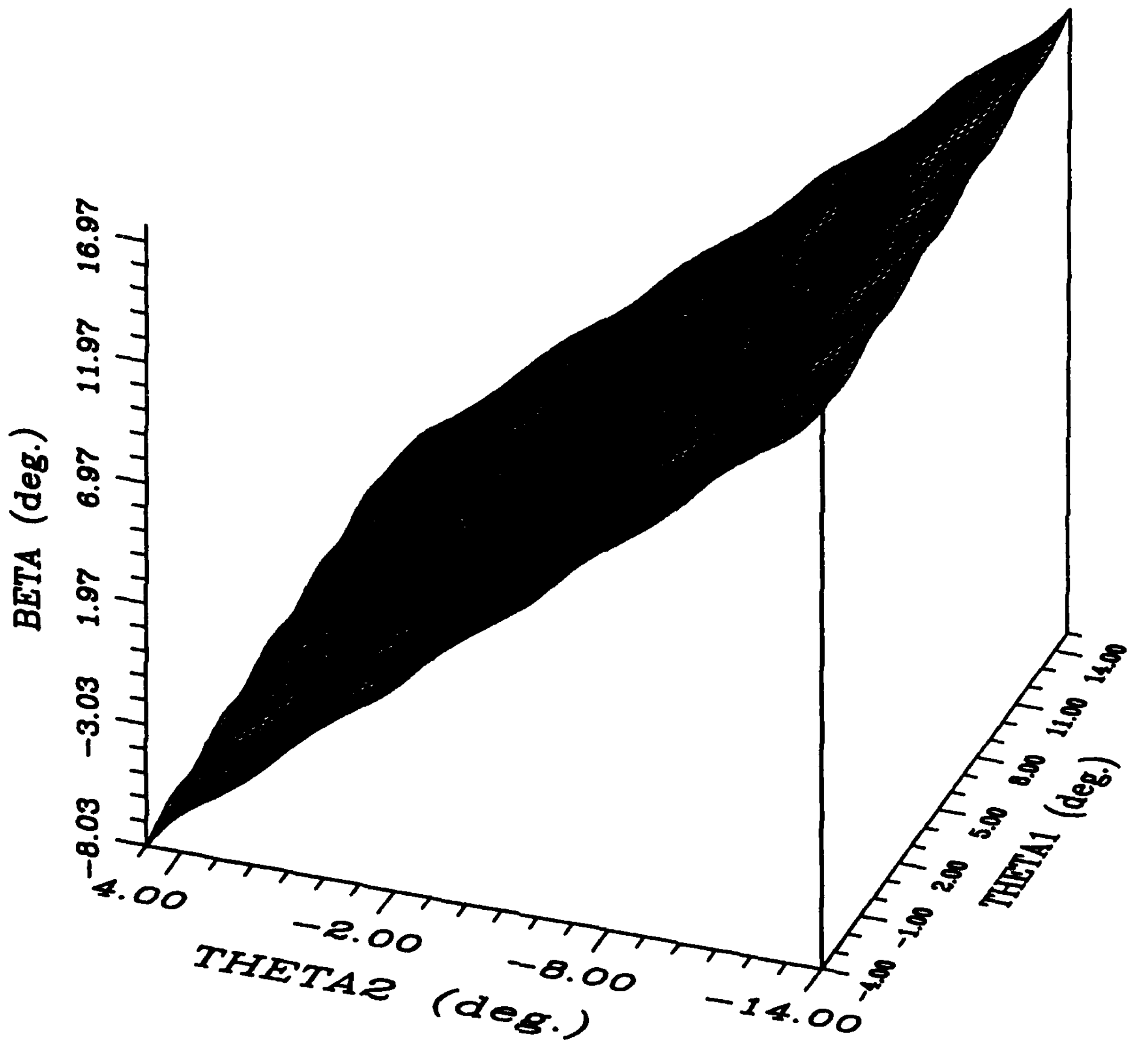
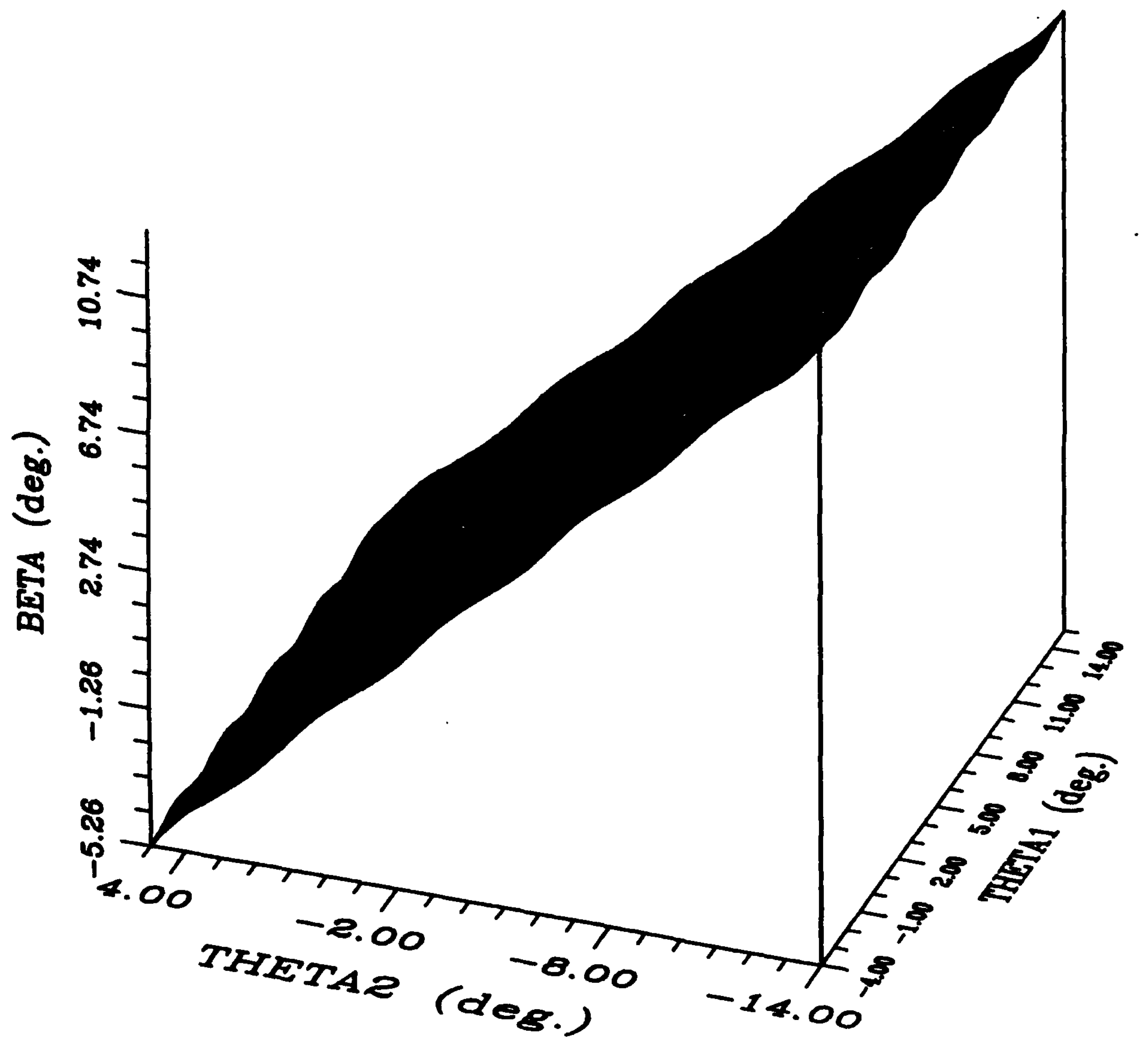


Figure 22b Closed-form rotor flapping angle β
$\alpha = -5$ deg. $\theta_0 = 15$ deg. $\Psi = 0$ deg.



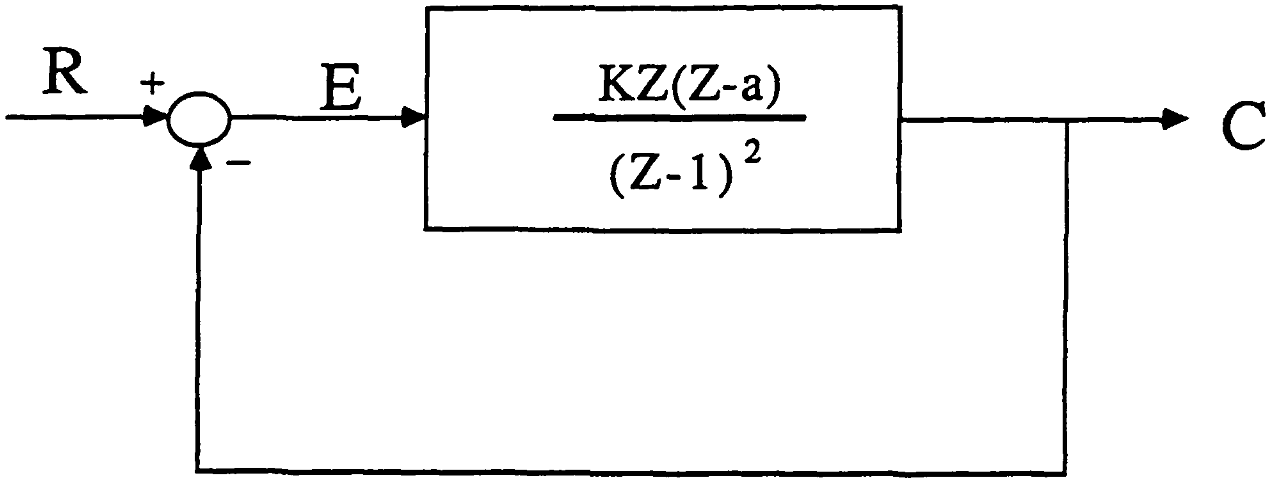


Figure 23a Block diagram of the tracking filter.

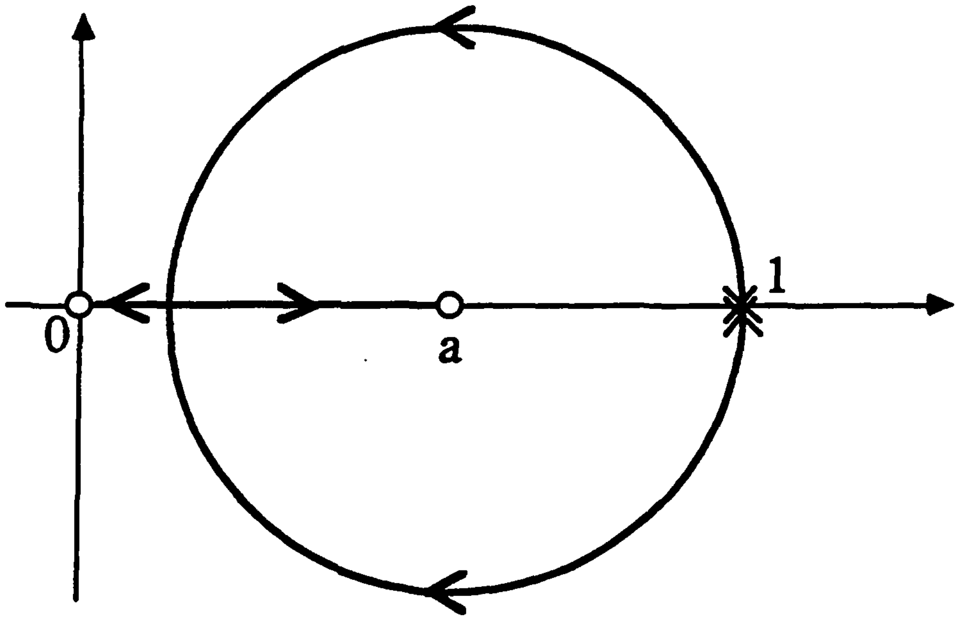


Figure 23b Root- locus of the tracking filter (z-plane) a double open-loop pole is located at z=1 and open-loop zeros at z=0 and z=a.

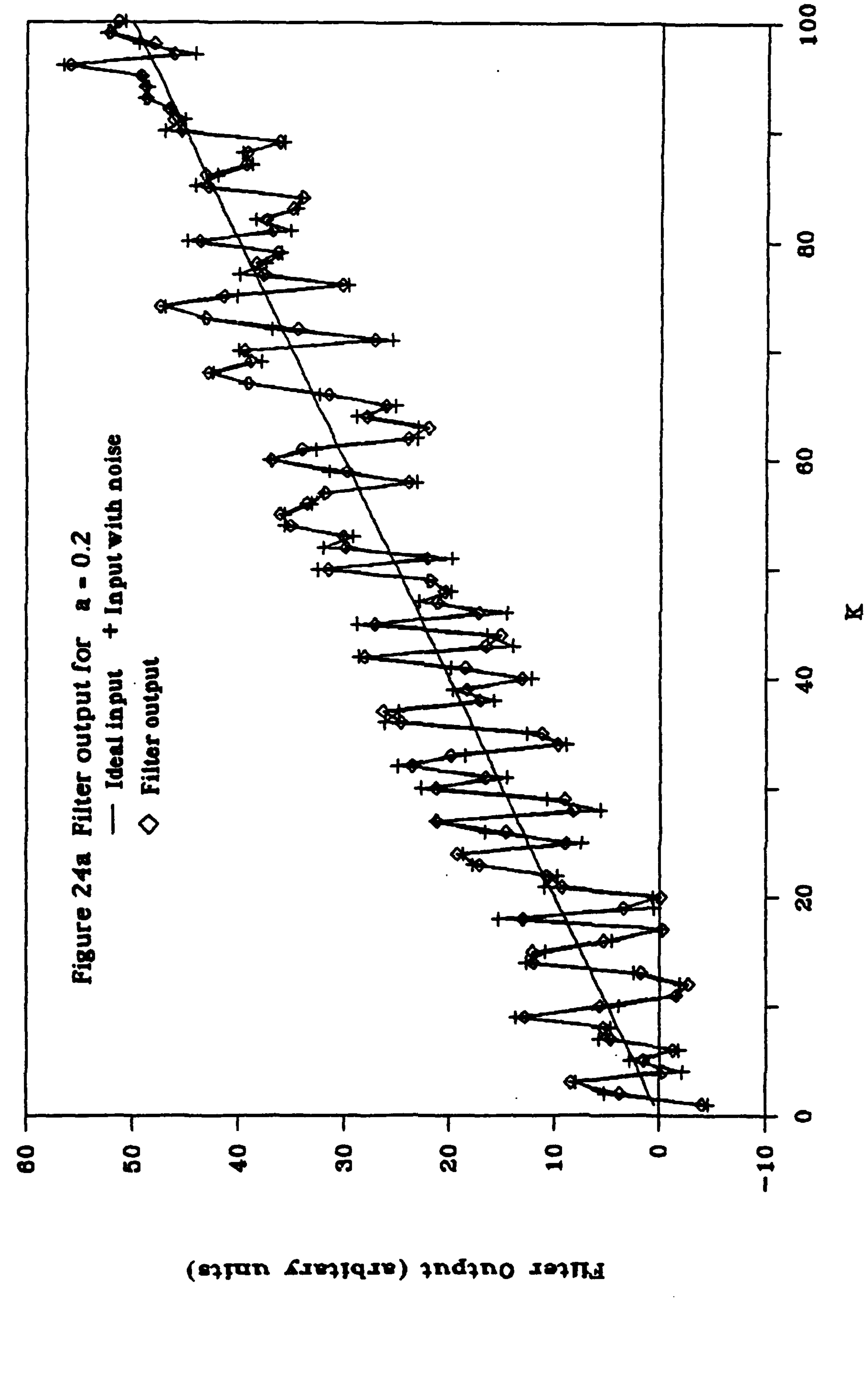


Figure 24a Filter output for a = 0.2
— Ideal input + Input with noise
◇ Filter output
Filter Output (arbitrary units)
K
60
50
40
30
20
10
0
−10
0
20
40
60
80
100

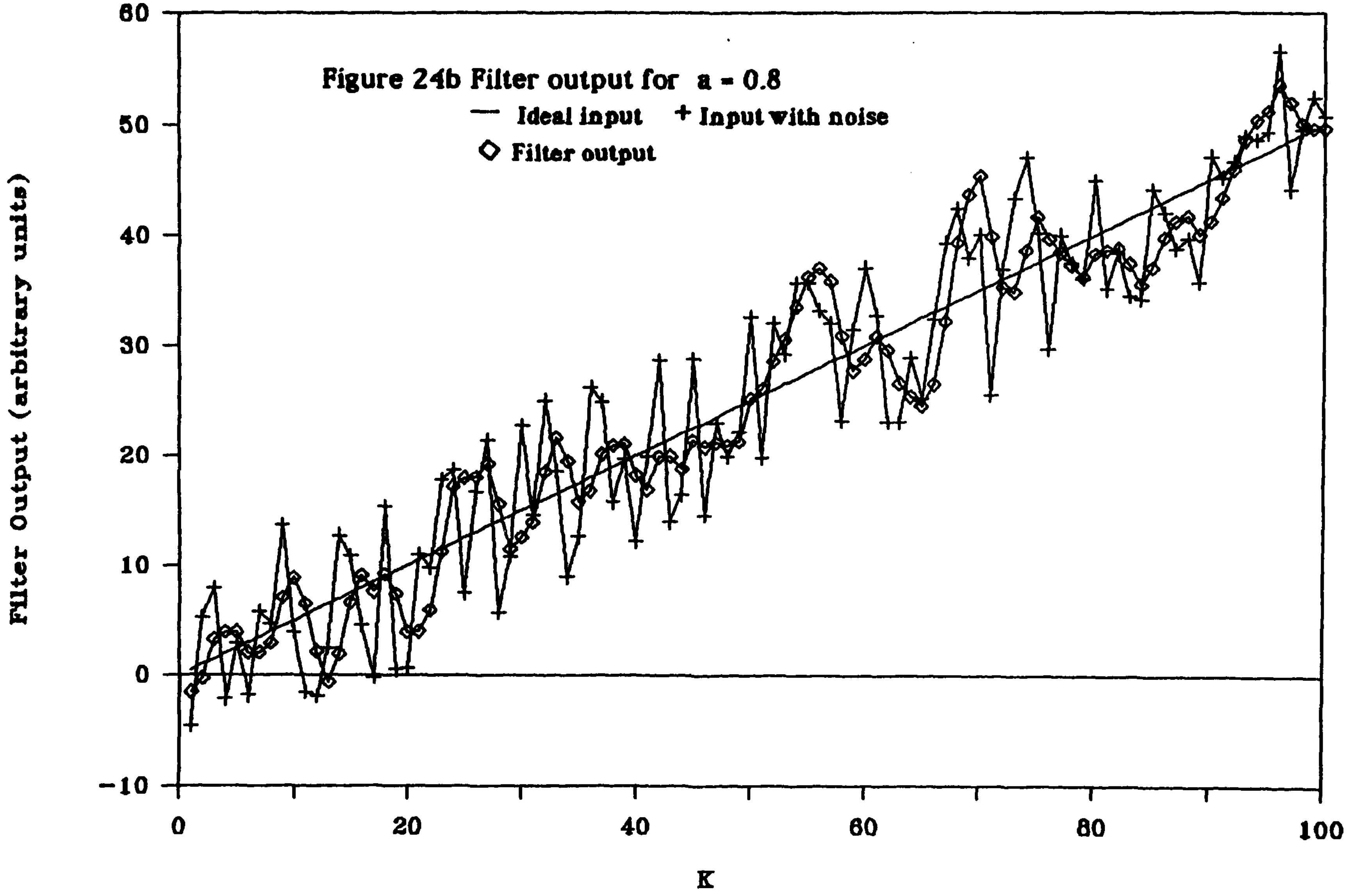


Figure 24b Filter output for a = 0.8

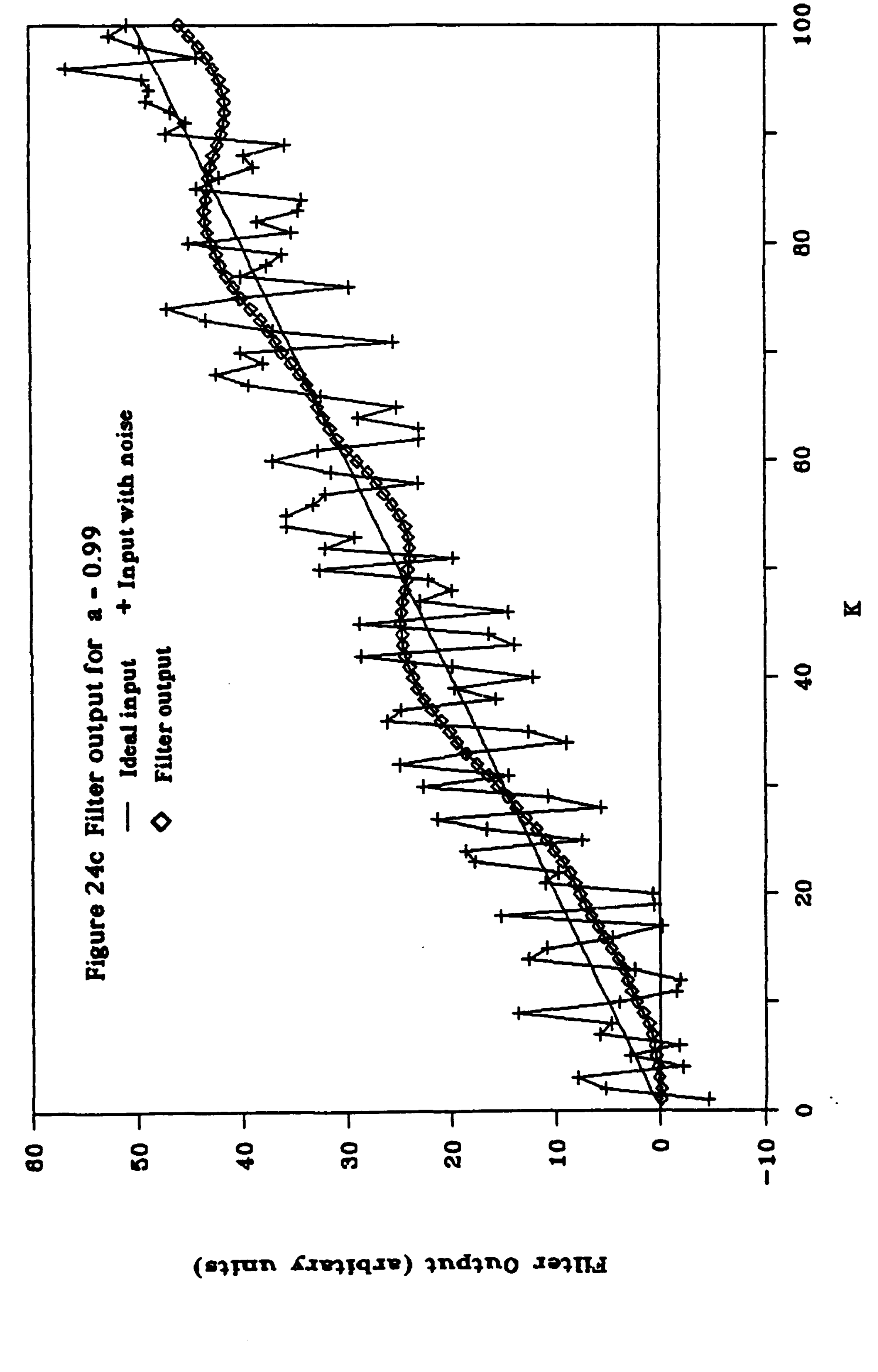


Figure 24c Filter output for a - 0.99

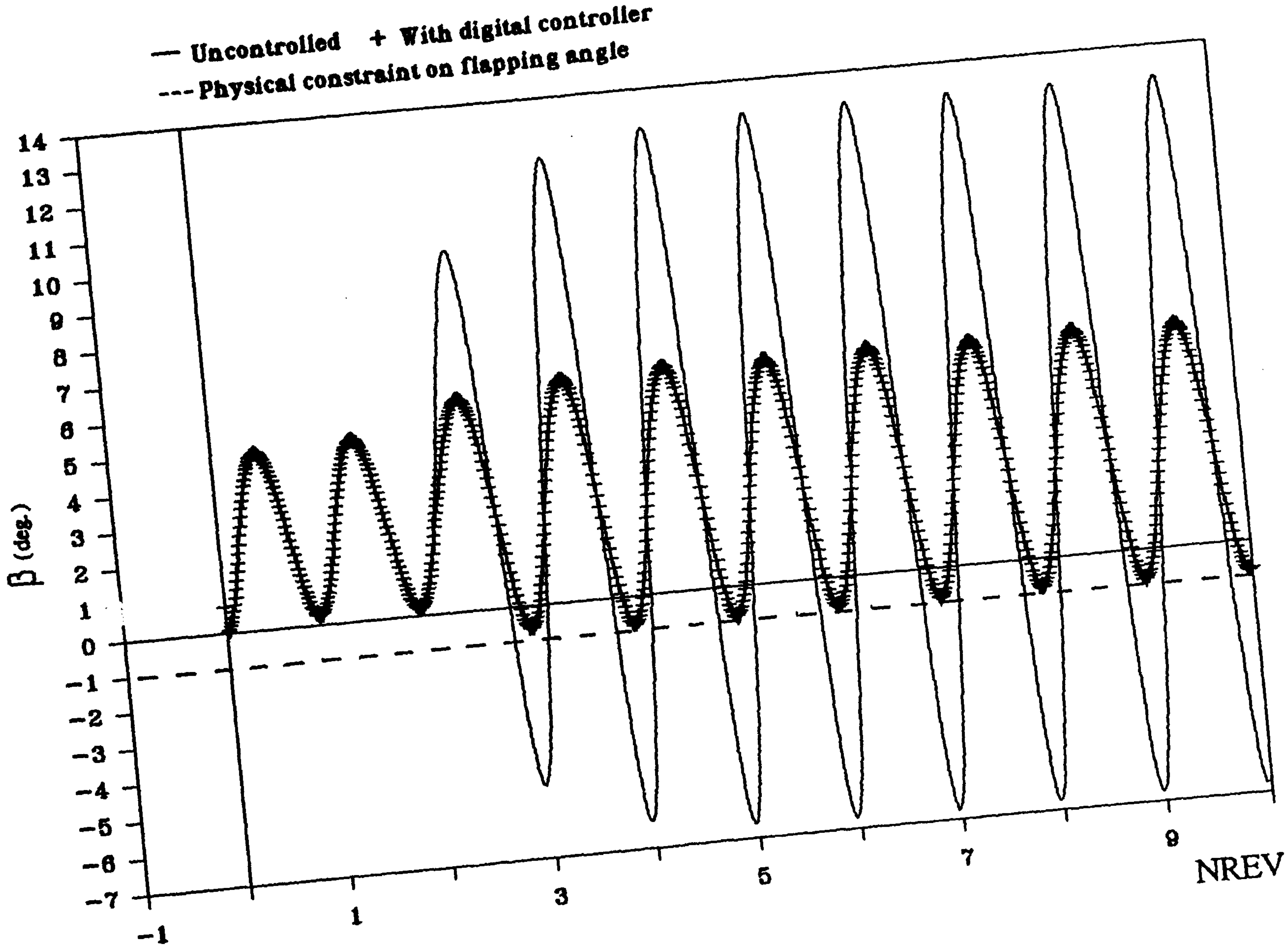


Figure 25a Rotor flapping response to step input of $\Delta\theta_1 = 6$ deg. at NREV = 2

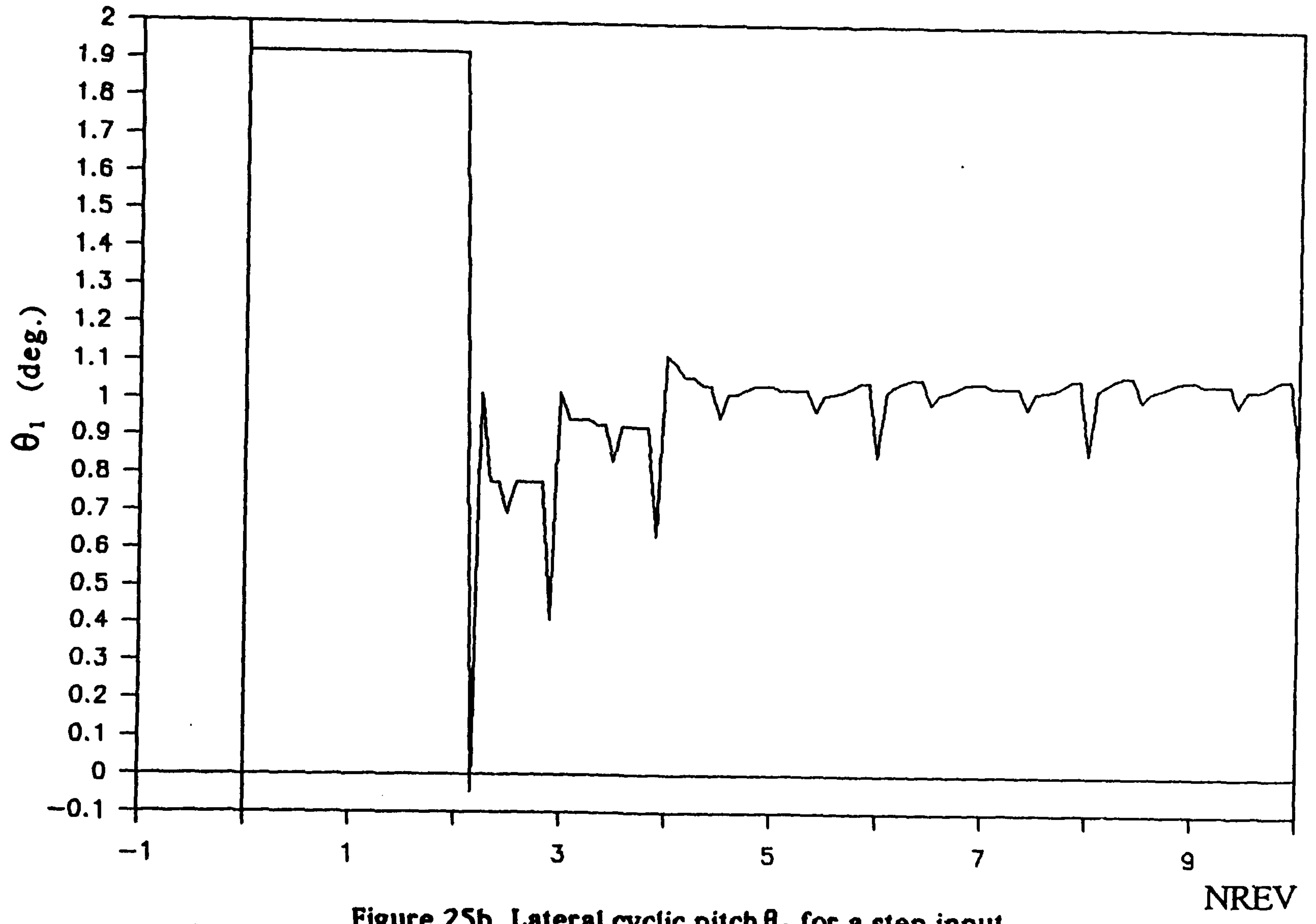


Figure 25b Lateral cyclic pitch θ_1 for a step input of $\Delta\theta_1 = 6$ deg. at NREV = 2 (with digital controller)

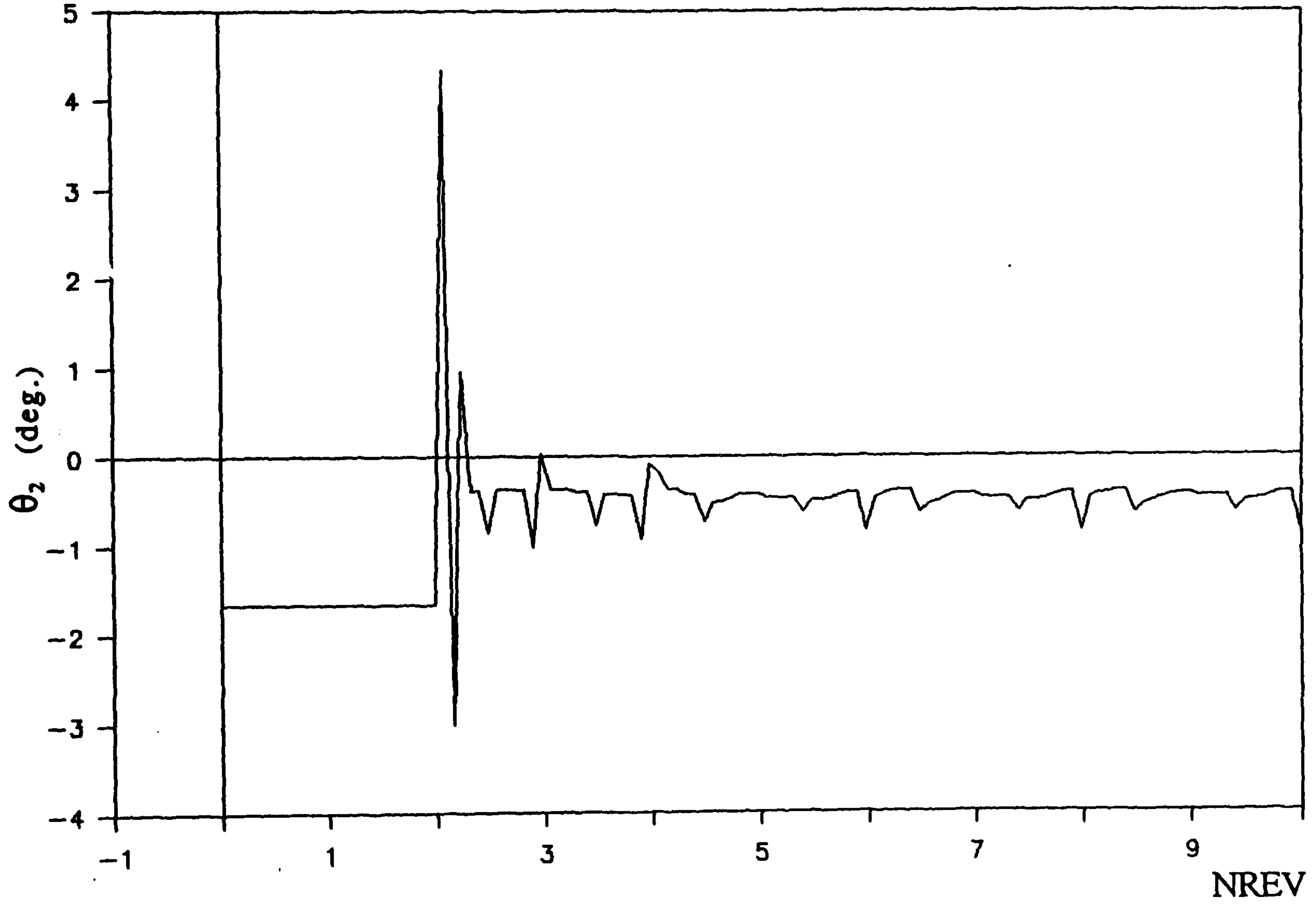


Figure 25c Longitudinal cyclic pitch θ_2 for a step input of $\Delta\theta_1 = 6$ deg. at NREV = 2

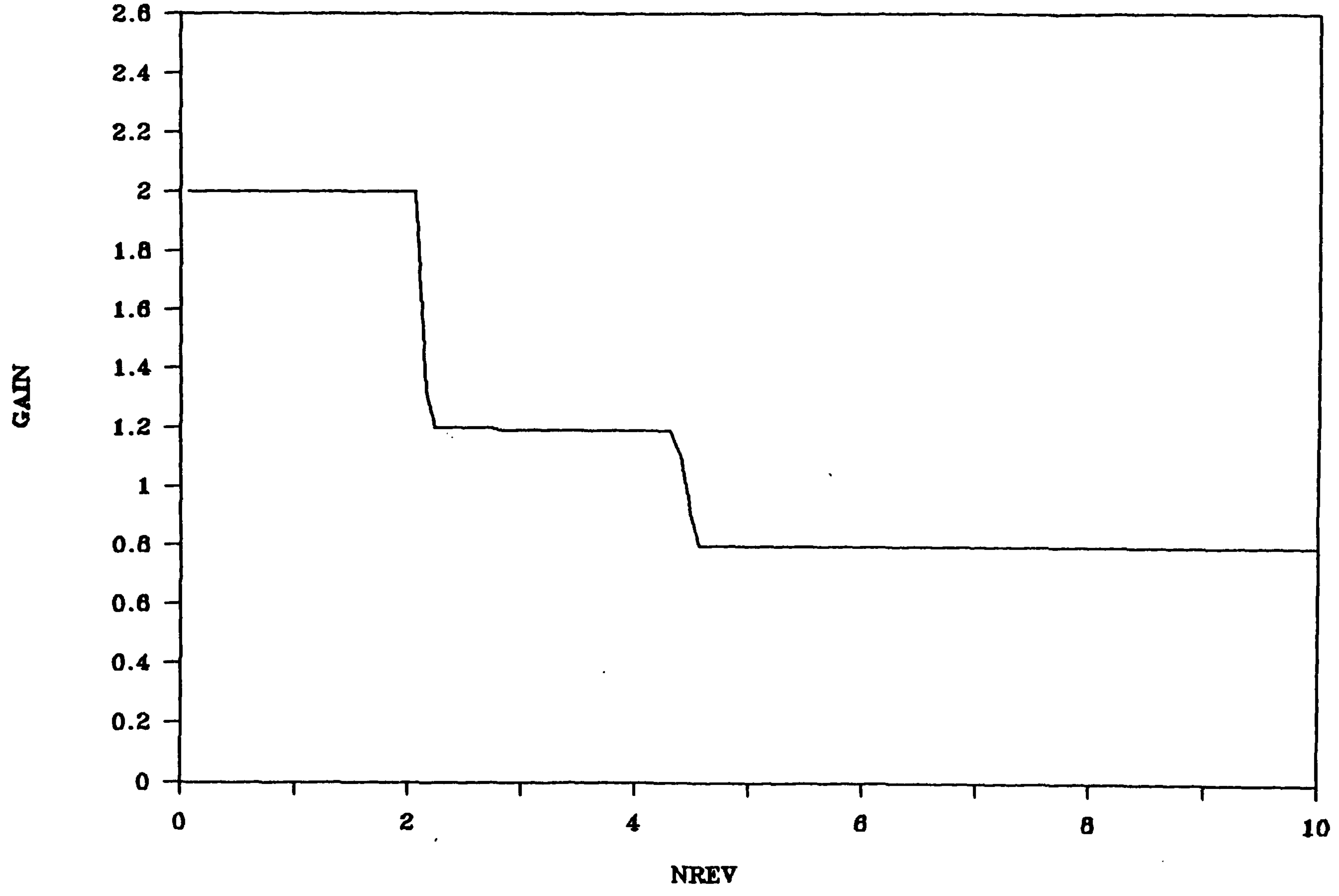


Figure 25d Feedback gain for a step input of $\Delta\theta_1 = 6$ deg, at NREV = 2

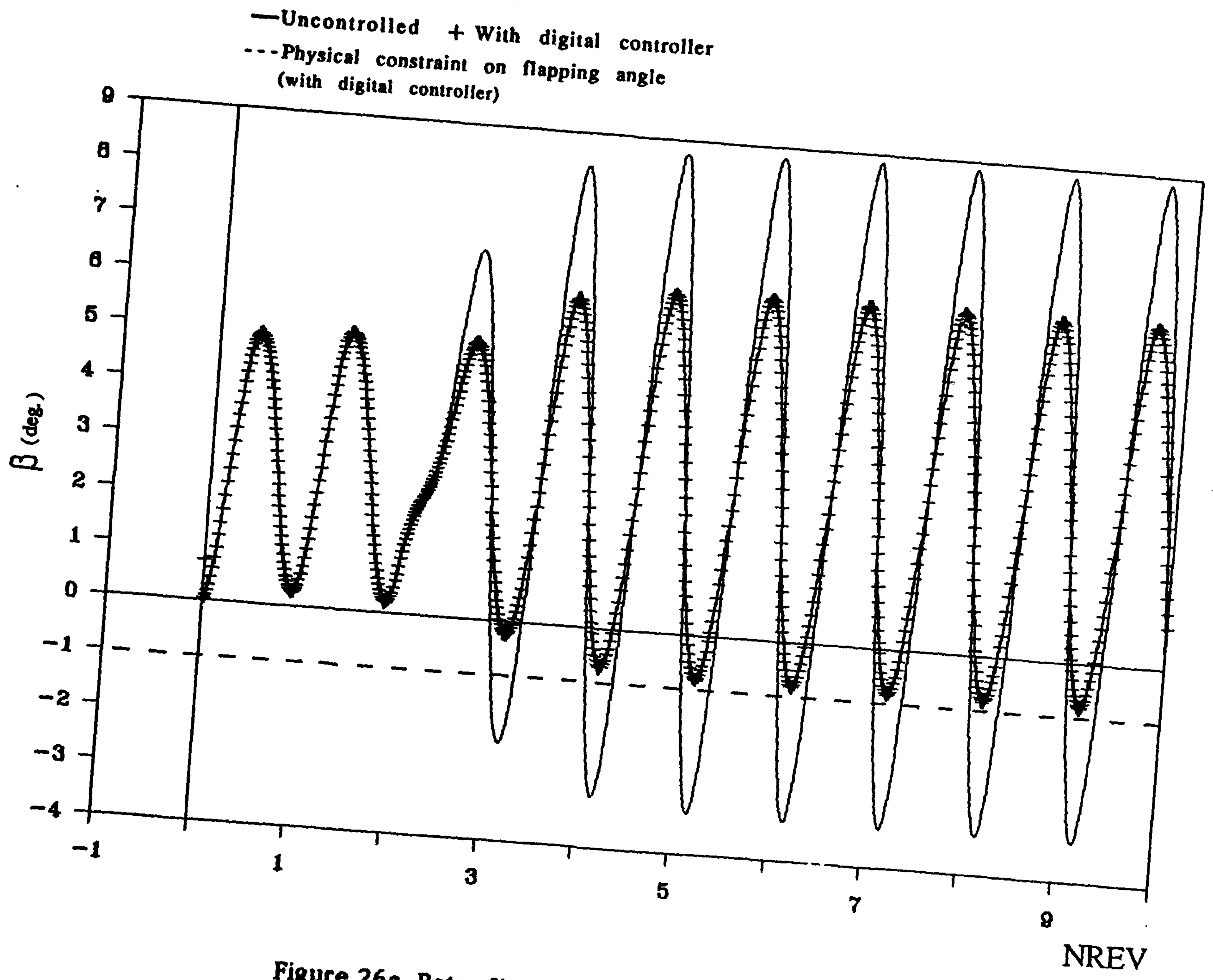


Figure 26a Rotor flapping response to step input of $\Delta\theta_2 = -6$ deg. at NREV = 2

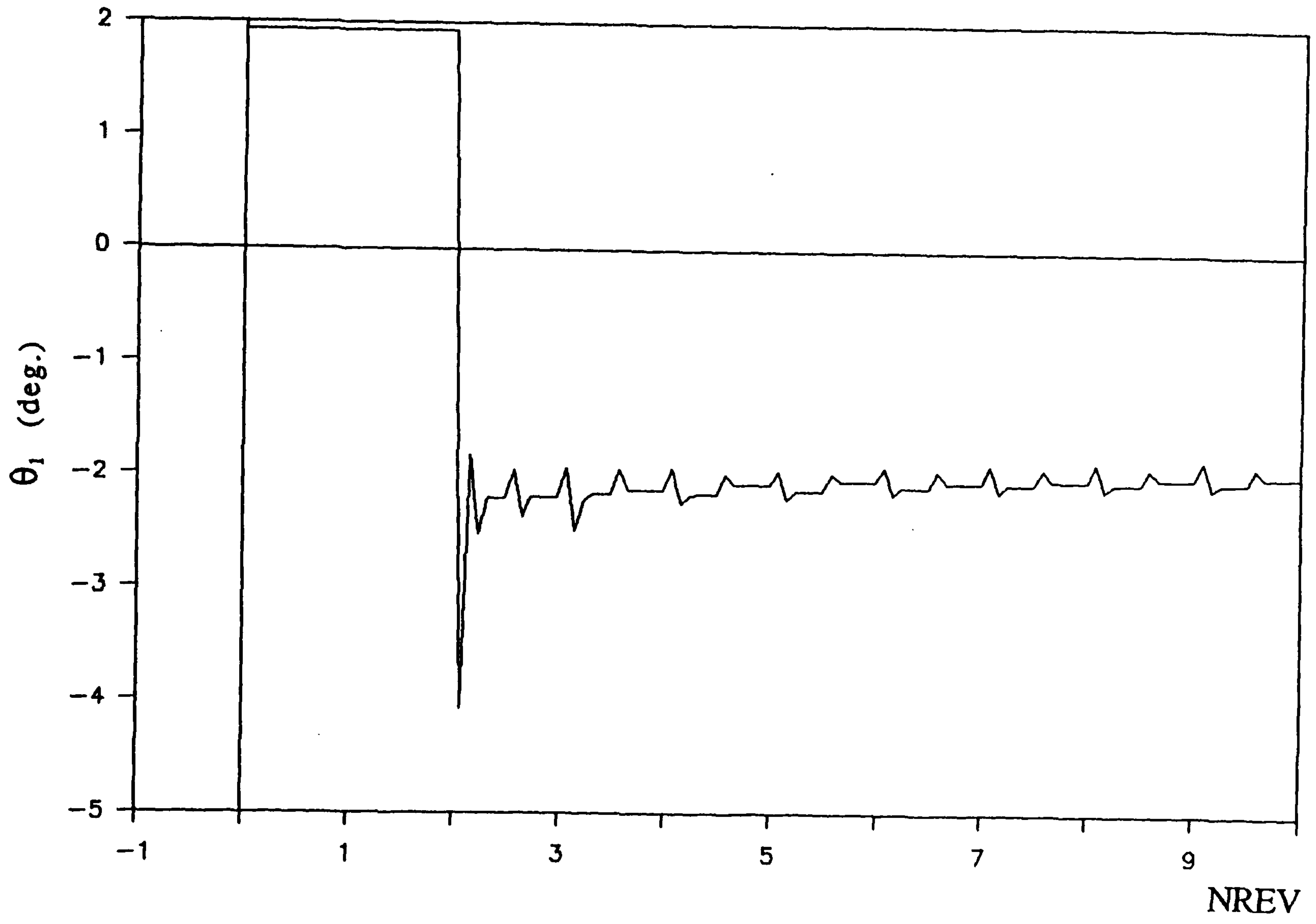


Figure 26b Lateral cyclic pitch θ_1 for a step input of $\Delta\theta_2$ = -6 deg. at NREV = 2 (with digital controller)

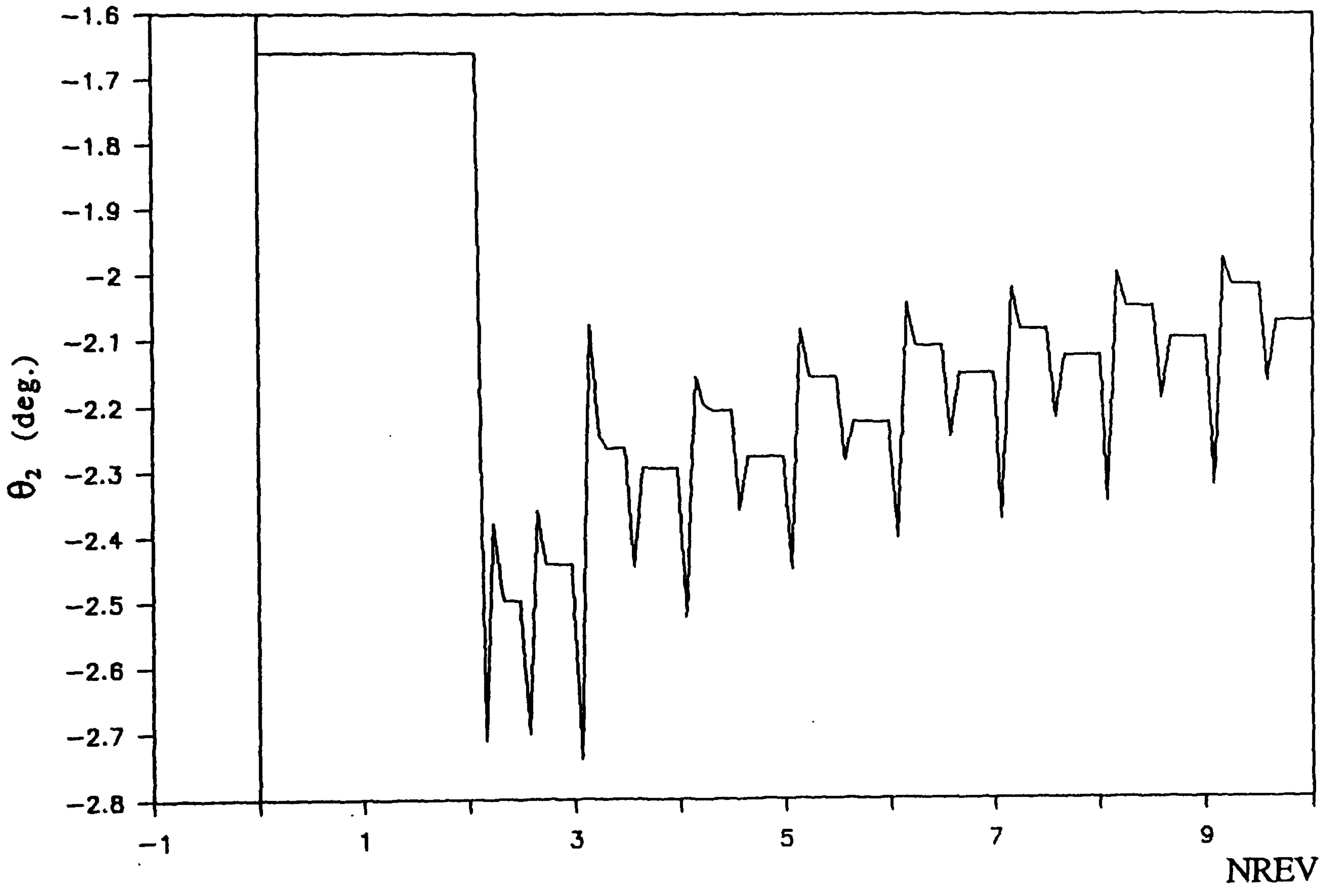


Figure 26c Longitudinal cyclic pitch θ_2 for a step input of $\Delta\theta_2 = -6$ deg. at NREV = 2
(with digital controller)

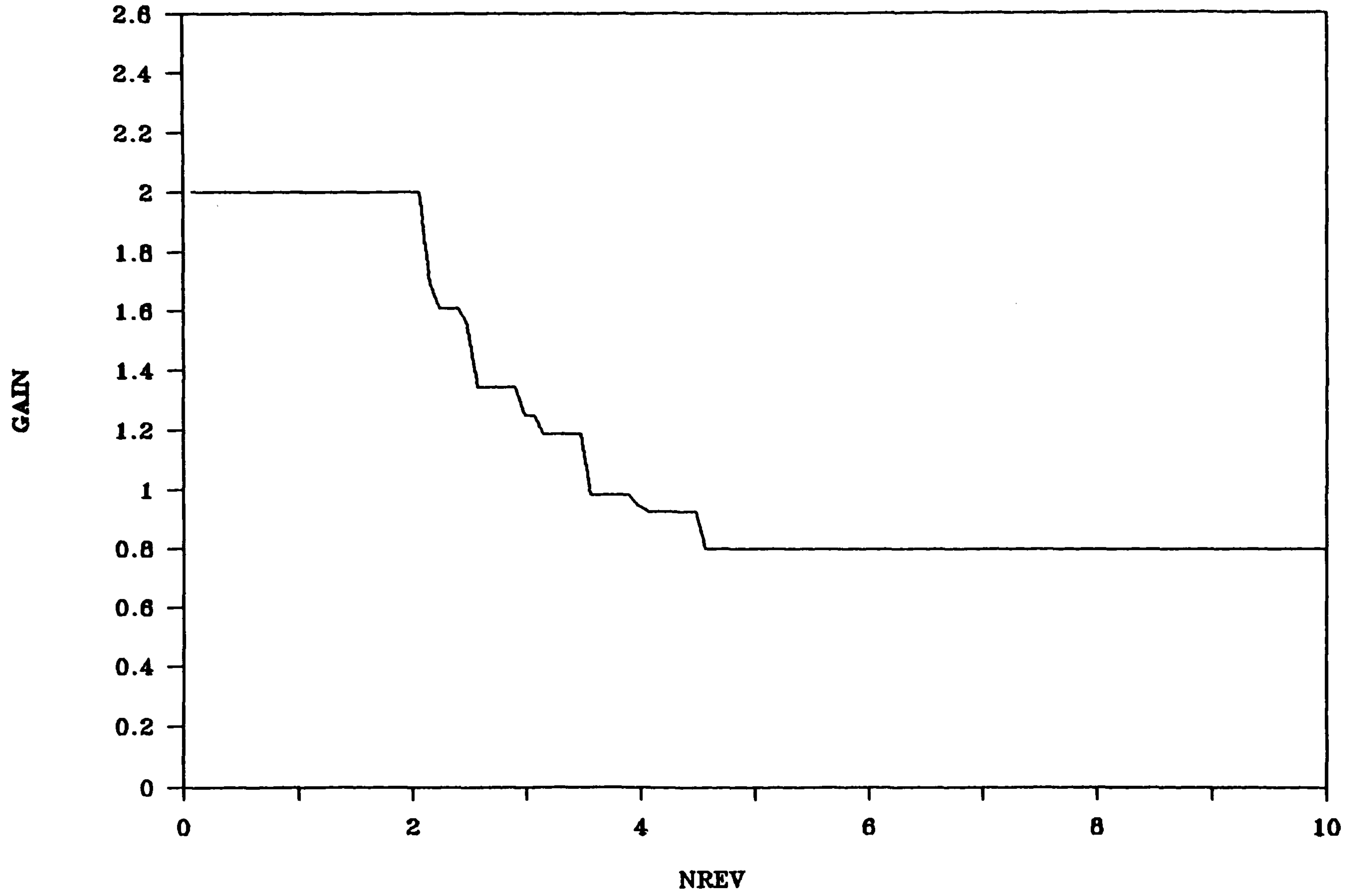


Figure 26d Feedback gain for a step input of $\Delta\theta_2$ = -6 deg. at NREV = 2

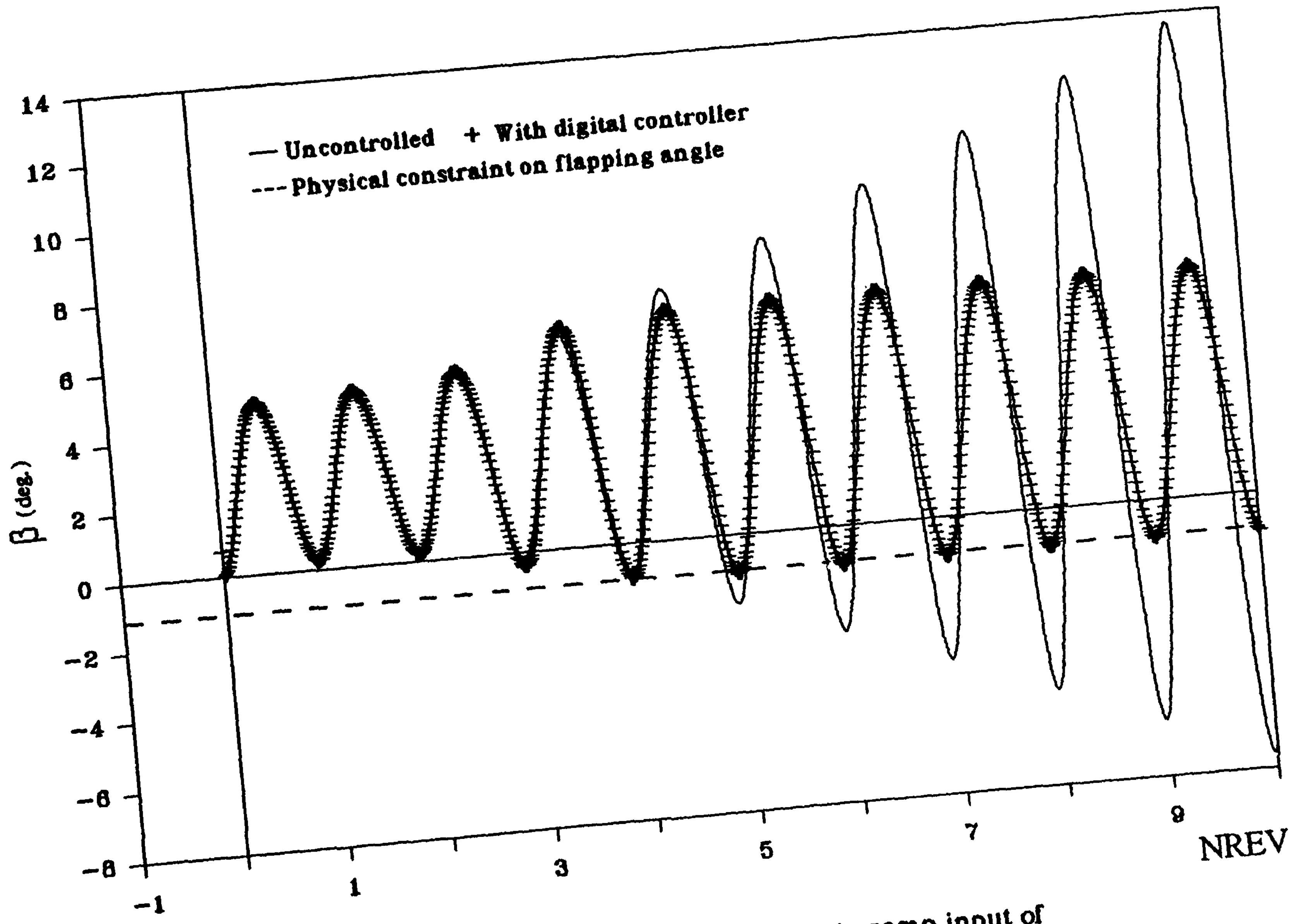


Figure 27a Rotor flapping response to ramp input of $\dot{\theta}_1 = 5$ deg/sec at NREV = 2

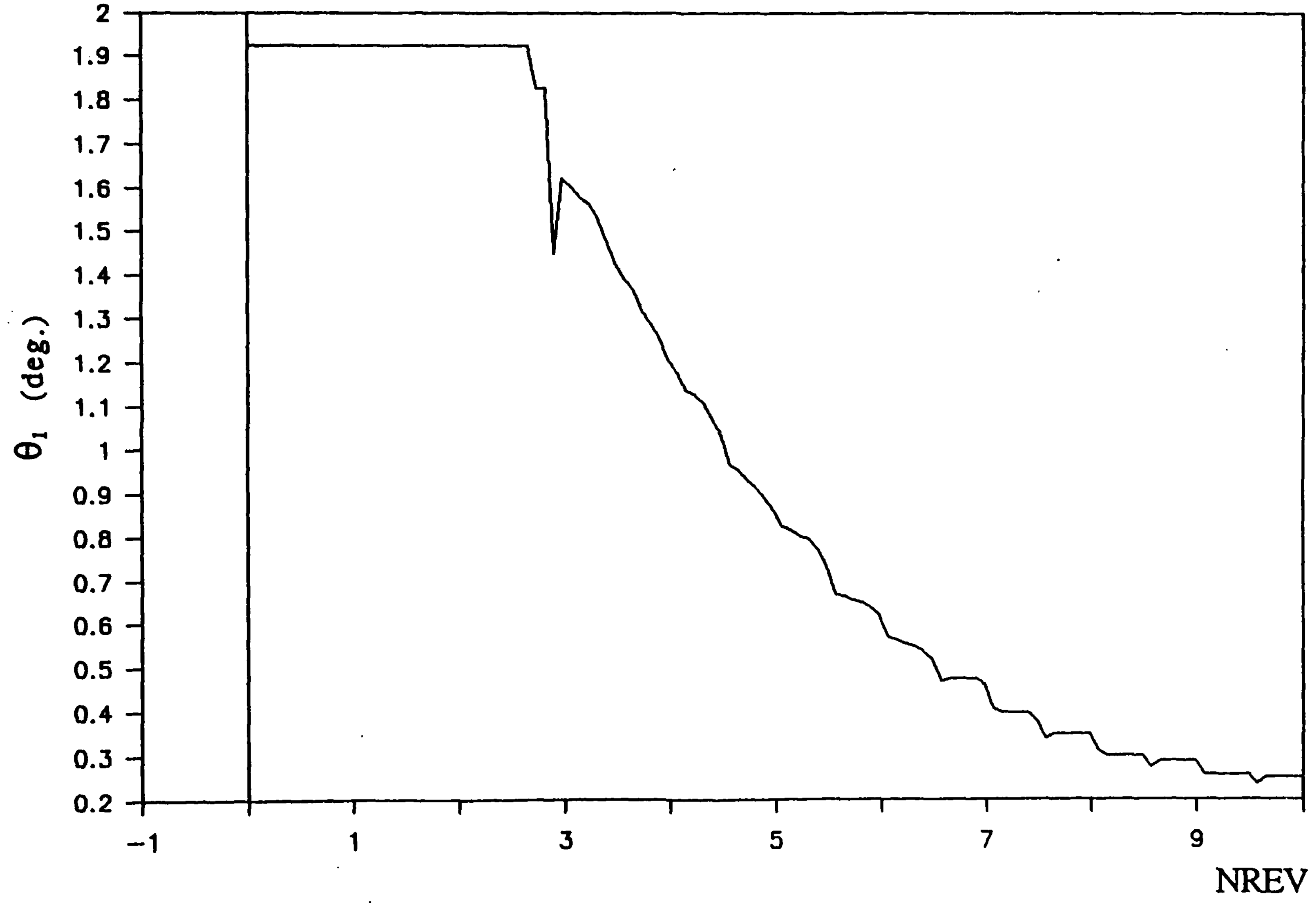


Figure 27b Lateral cyclic pitch θ_1 for a ramp input of $\dot{\theta}_1 = 5$ deg/sec at NREV = 2 (with digital controller)

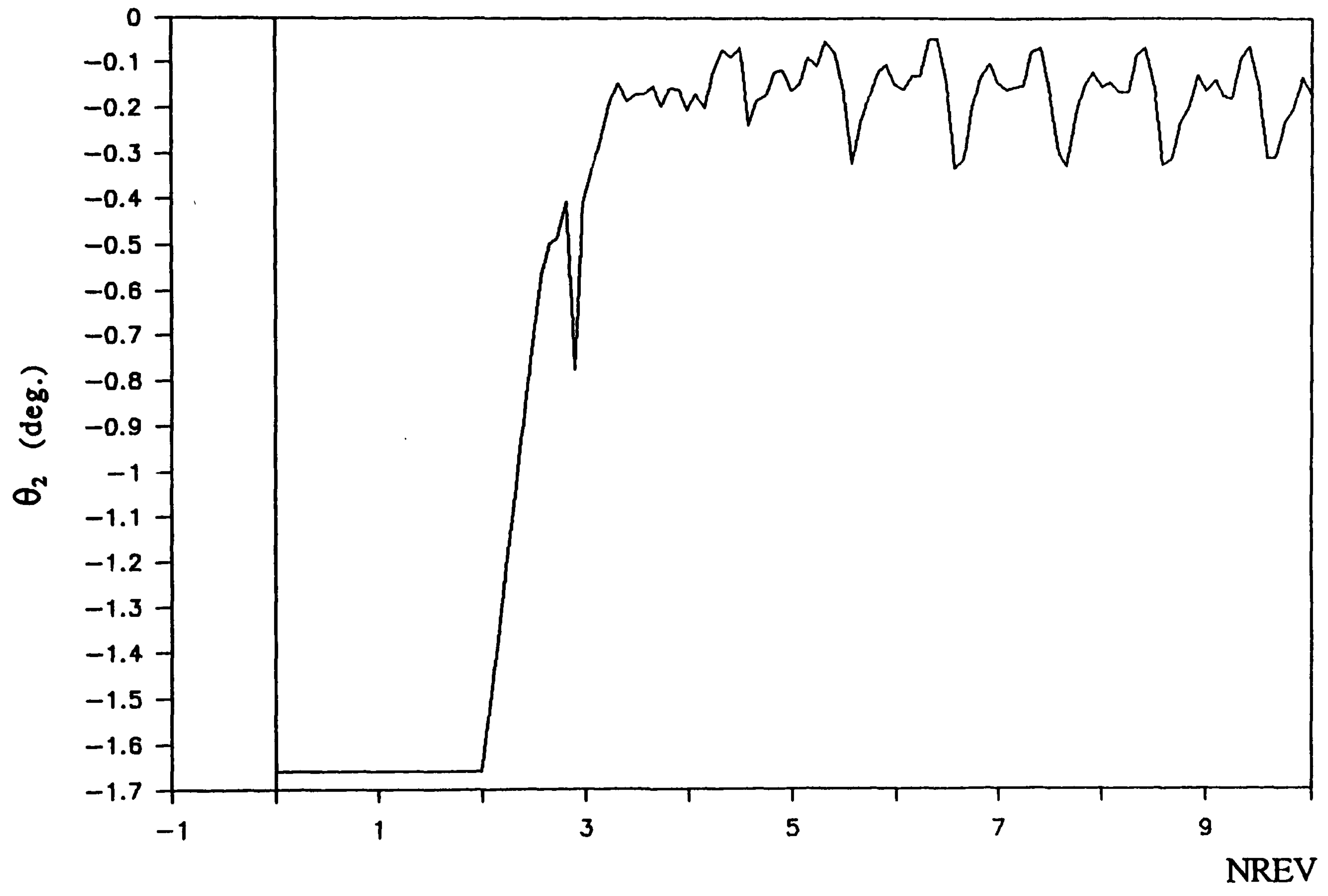


Figure 27c Longitudinal cyclic pitch θ_2 for a ramp input of $\dot{\theta}_1 = 5$ deg/sec at NREV = 2 (with digital controller)

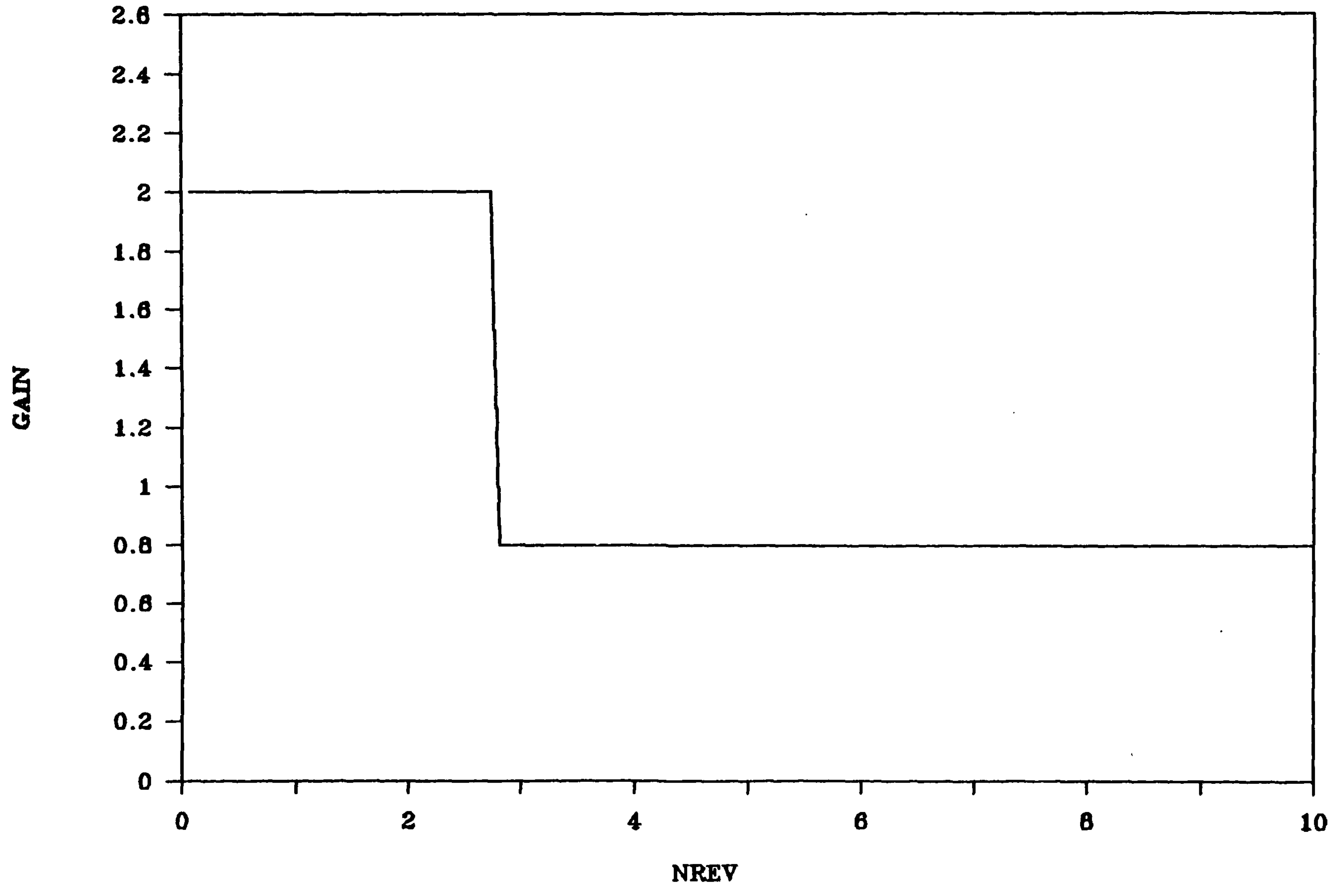


Figure 27d Feedback gain for ramp input of $\dot{\theta}_1 = 5$ deg/sec at NREV = 2

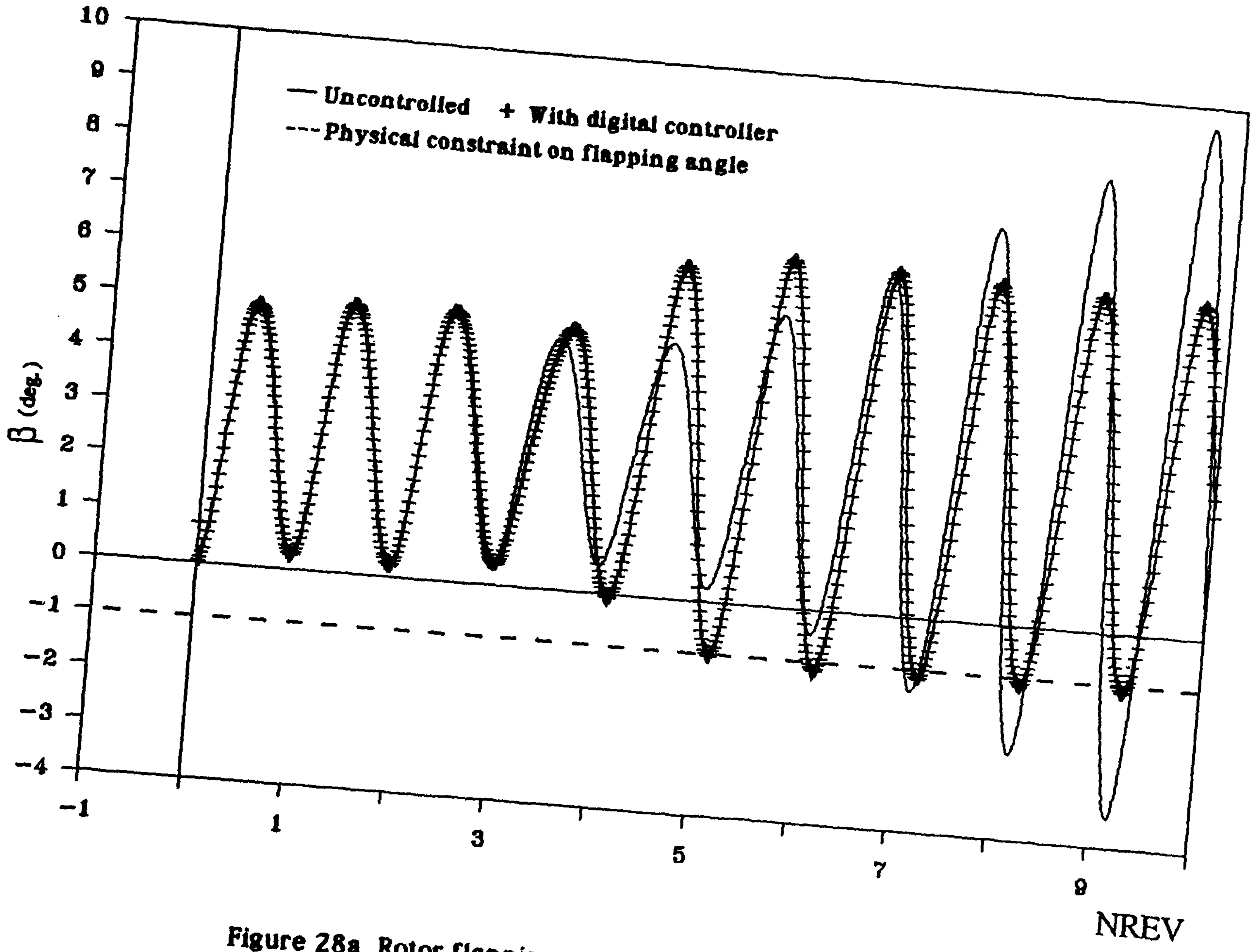


Figure 28a Rotor flapping response to a ramp input of $\dot{\theta}_2 = 5$ deg/sec at NREV = 2

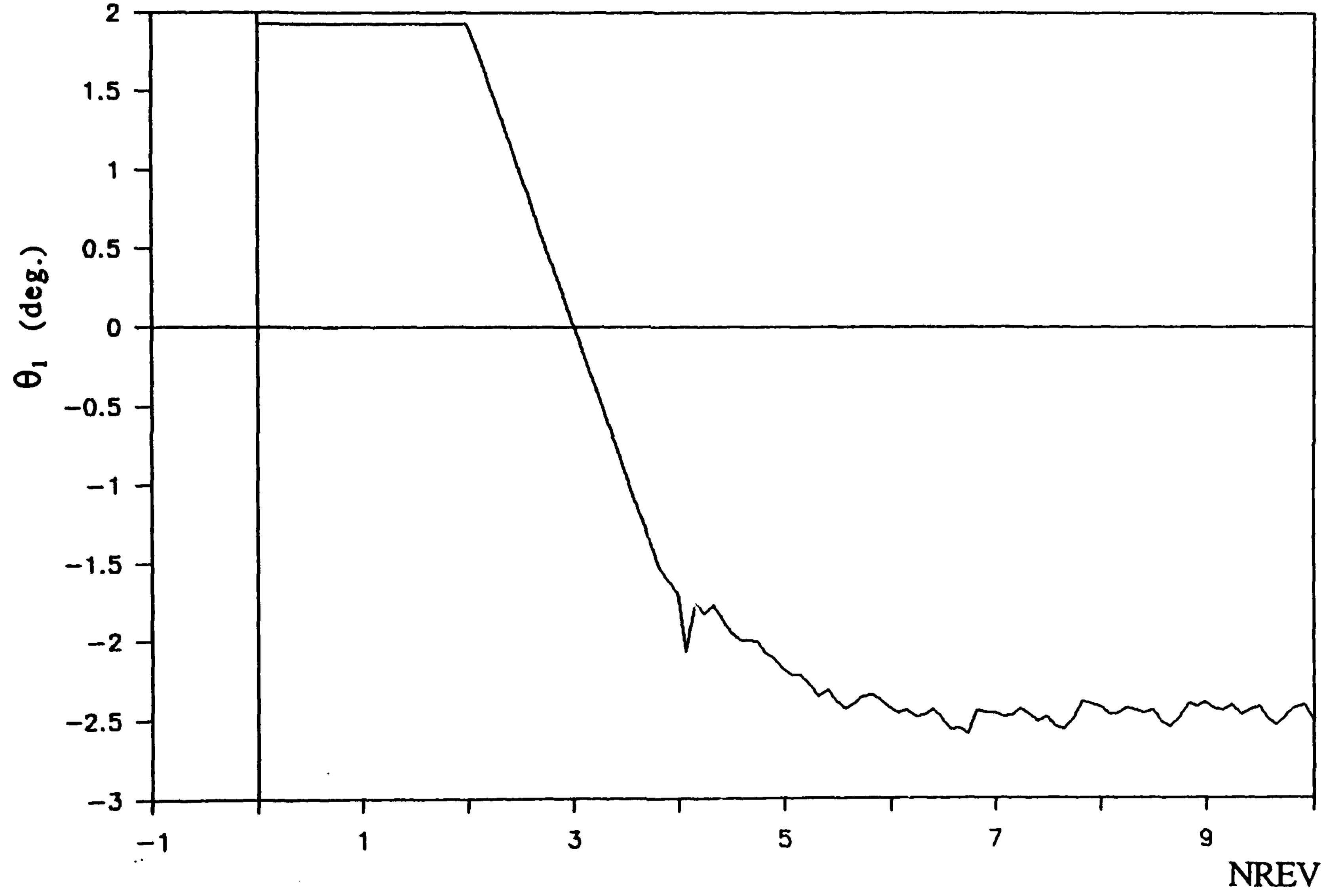


Figure 28b Lateral cyclic pitch θ_1 for a ramp input
$\dot{\theta}_1$ = 5 deg/sec at NREV = 2
(with digital controller)

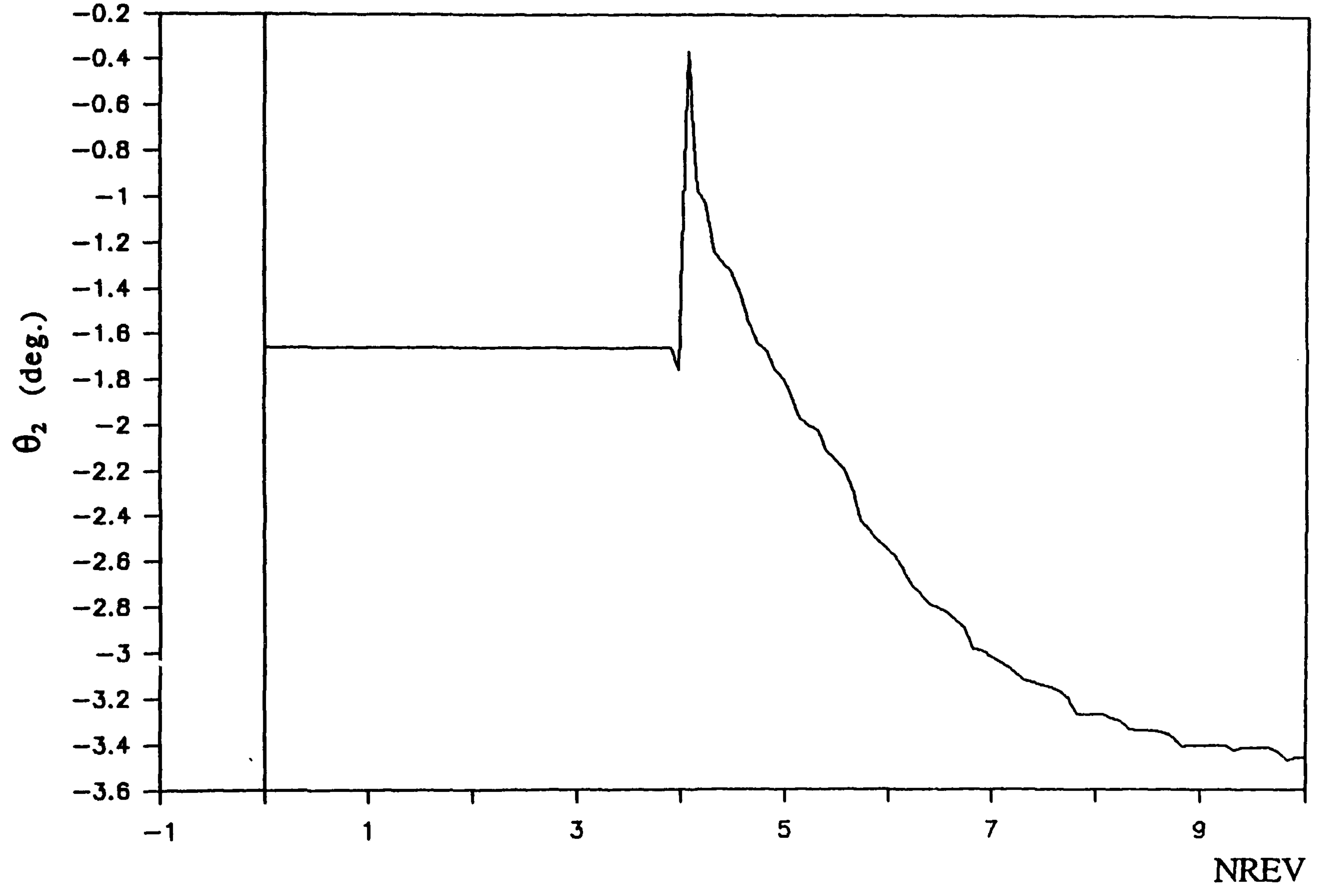


Figure 28c Longitudinal cyclic pitch θ_2 for a ramp input of $\dot{\theta}_2 = 5$ deg/sec at NREV = 2

(with digital controller)

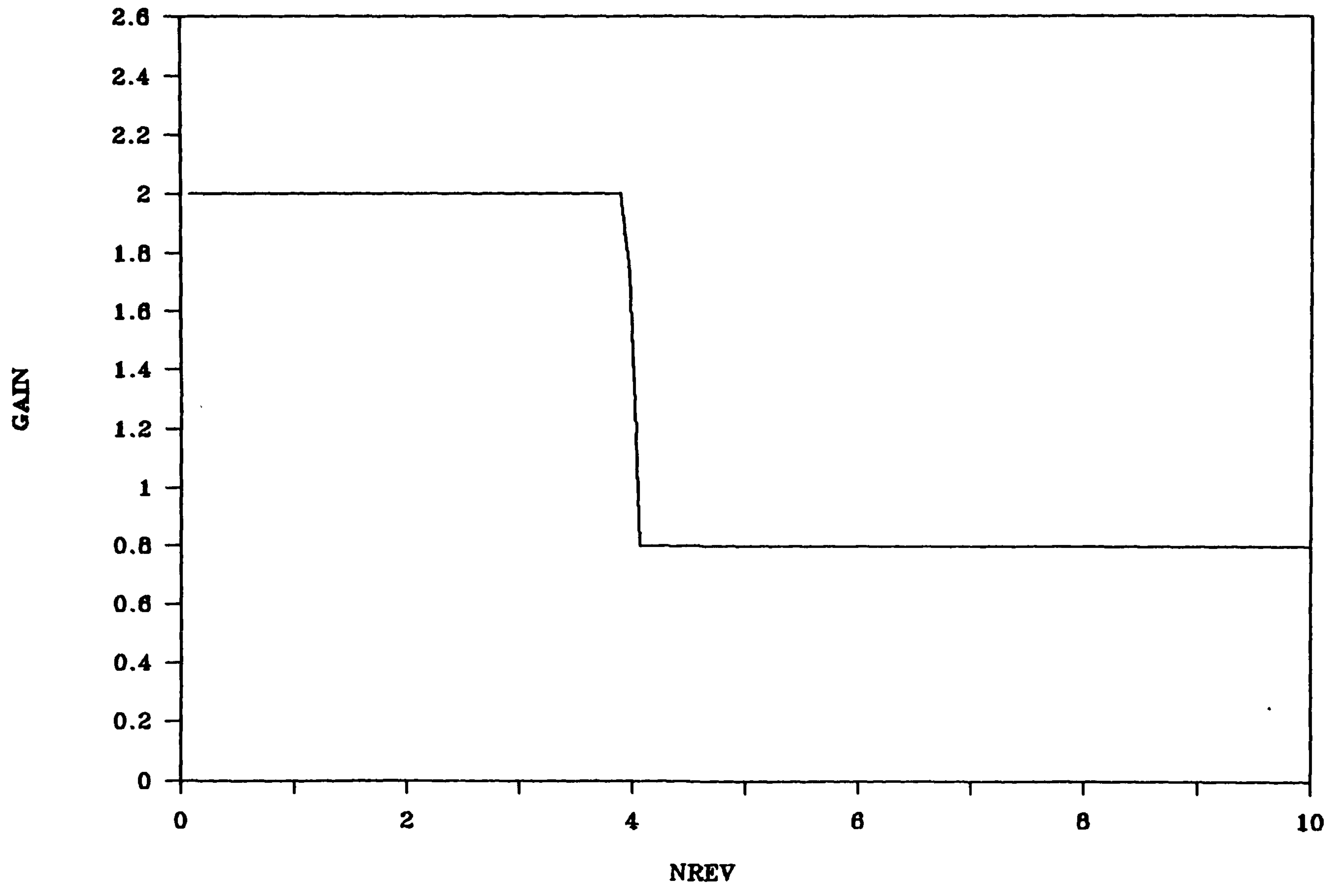


Figure 28d Feedback gain for a ramp input of
$\dot{\theta}_2$ = 5 deg/sec at NREV = 2

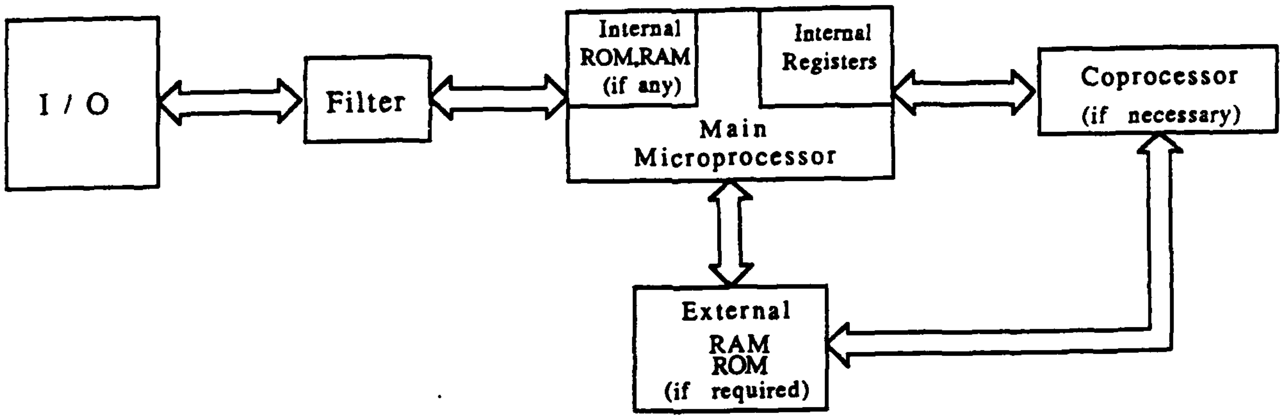

I / O
Filter
Internal ROM,RAM (if any)
Internal Registers
Main Microprocessor
Coprocessor (if necessary)
External RAM ROM (if required)

Figure 29 Block diagram of the microprocessor controller.

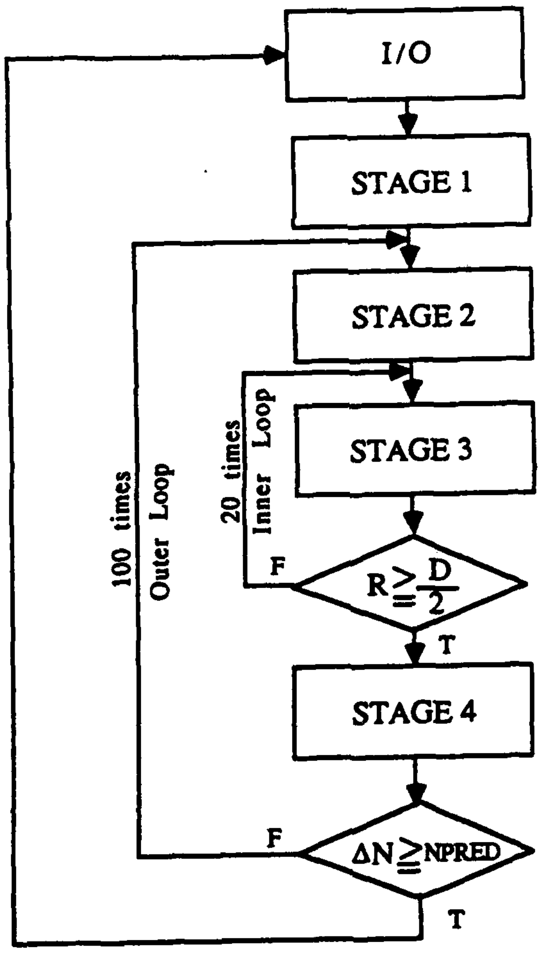


Figure 30 Software flowchart for analyzing microprocessor operations.

FORTRAN listing for PROGRAM FLAP
(numerical controller)

```
       PROGRAM FLAP
       COMMON/COM1/THE1LM,THE2LM,KNTRL,TCON,TOFF,THE1I,THE2I,SWITCH
       COMMON/COM2/DELFB1,DELFB2,FB,PI,CRATE1,CRATE2,FDBK1,FDBK2
       COMMON/COM3/CLI(20),ALPHAL(20),CDI(33),ALPHAD(33),ALPHAB,CL,CD
       COMMON/COM4/OMEGA,V,ALPHA,W,BETAD,RHO,DELPSI,DELR,T,MIF,MW,D
       COMMON/COM5/PSI,EPS,CO,CT,THETA0,THETAT,THETA1,THETA2,KBETA,BETA
       COMMON/COM6/FB1LIM,FB2LIM
       REAL MIF,MW,KBETA,N,NPRED,NMAX
       OPEN(UNIT=1,FILE='CL.DAT')
       DO 100 I=1,20
       READ(1,*) ALPHAL(I),CLI(I)
100    CONTINUE
       CLOSE (UNIT=1)
       OPEN(UNIT=2,FILE='CD.DAT')
       DO 200 I=1,33
       READ(2,*) ALPHAD(I),CDI(I)
200    CONTINUE
       CLOSE(UNIT=2)
       PI=3.14159
       TWOPI=2.*PI
       DTR=PI/180.
       DO 300 I=1,20
       ALPHAL(I)=ALPHAL(I)*DTR
300    CONTINUE
       DO 400 I=1,33
       ALPHAD(I)=ALPHAD(I)*DTR
400    CONTINUE
       WRITE(*,*)'THIS PROGRAM PREDICTS THE FLAPPING MOTION FOR A'
       WRITE(*,*)'HELICOPTER ROTOR BLADE IN FORWARD FLIGHT WITH PILOT'
       WRITE(*,*)'INPUT AND FEEDBACK CONTROL.'
       WRITE(*,*)
       WRITE(*,*)
       WRITE(*,*)'INPUT IDENTIFYING CASE NUMBER'
       READ(*,*)CASE
       WRITE(*,*)'INPUT THE MAXIMUM FLAPPING ANGLE IN DEGREES WHICH '
       WRITE(*,*)'WILL BE ALLOWED'
       READ(*,*)BETLIM
       WRITE(*,*)'INPUT THE INCREMENTAL CORRECTION TO LATERAL CYCLIC'
       WRITE(*,*)'PITCH CONTROL WHICH IS ADDED EACH TIME BETA IS'
       WRITE(*,*)'PREDICTED TO EXCEED THE LIMIT.  IT IS ALSO SUBTRACTED'
       WRITE(*,*)'EACH TIME BETA IS PREDICTED NOT TO EXCEED THE LIMIT'
       WRITE(*,*)'IF ANY FEEDBACK IS BEING APPLIED.'
       READ(*,*)DELFB1
       WRITE(*,*)'SIMILARLY, INPUT THE INCREMENTAL CORRECTION TO'
       WRITE(*,*)'THE LONGITUDINAL CYCLIC'
       READ(*,*)DELFB2
       WRITE(*,*)'INPUT THE LIMIT ON FEEDBACK TO LATERAL CYCLIC,DEGS.'
       READ(*,*)FB1LIM
       WRITE(*,*)'INPUT THE LIMIT ON FEEDBACK TO LONG. CYCLIC, DEGS.'
       READ(*,*)FB2LIM
       WRITE(*,*)'INPUT NUMBER OF ROTATIONS BEFORE INITIATING CONTROL'
       READ(*,*)N
       WRITE(*,*)'INPUT NUMBER OF ROTATIONS TO PREDICT AHEAD'
       READ(*,*)NPRED
```

```
        PROGRAM FLAP
        COMMON/COM1/THE1LM,THE2LM,KNTRL,TCON,TOFF,THE1I,THE2I,SWITCH
        COMMON/COM2/DELFB1,DELFB2,FB,PI,CRATE1,CRATE2,FDBK1,FDBK2
        COMMON/COM3/CLI(20),ALPHAL(20),CDI(33),ALPHAD(33),ALPHAB,CL,CD
        COMMON/COM4/OMEGA,V,ALPHA,W,BETAD,RHO,DELPSI,DELR,T,MIF,MW,D
        COMMON/COM5/PSI,EPS,CO,CT,THETA0,THETAT,THETA1,THETA2,KBETA,BETA
        COMMON/COM6/FB1LIM,FB2LIM
        REAL MIF,MW,KBETA,N,NPRED,NMAX
        OPEN(UNIT=1,FILE='CL.DAT')
        DO 100 I=1,20
        READ(1,*) ALPHAL(I),CLI(I)
100     CONTINUE
        CLOSE (UNIT=1)
        OPEN(UNIT=2,FILE='CD.DAT')
        DO 200 I=1,33
        READ(2,*) ALPHAD(I),CDI(I)
200     CONTINUE
        CLOSE(UNIT=2)
        PI=3.14159
        TWOPI=2.*PI
        DTR=PI/180.
        DO 300 I=1,20
        ALPHAL(I)=ALPHAL(I)*DTR
300     CONTINUE
        DO 400 I=1,33
        ALPHAD(I)=ALPHAD(I)*DTR
400     CONTINUE
        WRITE(*,*)'THIS PROGRAM PREDICTS THE FLAPPING MOTION FOR A'
        WRITE(*,*)'HELICOPTER ROTOR BLADE IN FORWARD FLIGHT WITH PILOT'
        WRITE(*,*)'INPUT AND FEEDBACK CONTROL.'
        WRITE(*,*)
        WRITE(*,*)
        WRITE(*,*)'INPUT IDENTIFYING CASE NUMBER'
        READ(*,*)CASE
        WRITE(*,*)'INPUT THE MAXIMUM FLAPPING ANGLE IN DEGREES WHICH '
        WRITE(*,*)'WILL BE ALLOWED'
        READ(*,*)BETLIM
        WRITE(*,*)'INPUT THE INCREMENTAL CORRECTION TO LATERAL CYCLIC'
        WRITE(*,*)'PITCH CONTROL WHICH IS ADDED EACH TIME BETA IS'
        WRITE(*,*)'PREDICTED TO EXCEED THE LIMIT.  IT IS ALSO SUBTRACTED'
        WRITE(*,*)'EACH TIME BETA IS PREDICTED NOT TO EXCEED THE LIMIT'
        WRITE(*,*)'IF ANY FEEDBACK IS BEING APPLIED.'
        READ(*,*)DELFB1
        WRITE(*,*)'SIMILARLY, INPUT THE INCREMENTAL CORRECTION TO'
        WRITE(*,*)'THE LONGITUDINAL CYCLIC'
        READ(*,*)DELFB2
        WRITE(*,*)'INPUT THE LIMIT ON FEEDBACK TO LATERAL CYCLIC DEGS.'
        READ(*,*)FB1LIM
        WRITE(*,*)'INPUT THE LIMIT ON FEEDBACK TO LONG. CYCLIC, DEGS.'
        READ(*,*)FB2LIM
        WRITE(*,*)'INPUT NUMBER OF ROTATIONS BEFORE INITIATING CONTROL'
        READ(*,*)N
        WRITE(*,*)'INPUT NUMBER OF ROTATIONS TO PREDICT AHEAD'
        READ(*,*)NPRED
```

```
      WRITE(*,*)'INPUT FRACTION OF REV REQUIRED TO PREDICT'
      WRITE(*,*)'BLADE MOTION NPRED REVOLUTIONS AHEAD'
      READ(*,*)FPRED
      WRITE(*,*)'ONLY CYCLIC CONTROLS WILL BE APPLIED. EACH CONTROL'
      WRITE(*,*)'WILL BE INCREASED LINEARLY UP TO A MAX VALUE.'
      WRITE(*,*)'INPUT: RATE OF INCREASE OF LATERAL CYCLIC, DEGS/SEC'
      WRITE(*,*)'       RATE OF INCREASE OF LONGITUDINAL CYC,DEGS/SEC'
      READ(*,*)CRATE1,CRATE2
      WRITE(*,*)'INPUT: MAX INCREMENTAL VALUE FOR LATERAL CONTROL, DEG'
      WRITE(*,*)'       MAX INCREMENTAL VALUE FOR LONG. CONTROL, DEGS'
      READ(*,*)THE1LM,THE2LM
      WRITE(*,*)'INPUT: LENGTH OF TIME FOR CONTROL TO BE APPLIED,SECS'
      WRITE(*,*)'       INPUT NUMBER OF CALCULATIONS BETWEEN PRINTOUTS'
      READ(*,*)TOFF,PRT
      WRITE(*,*)'INPUT MAX NUMBER OF REVS FOR RUN'
      READ(*,*)NMAX
      OPEN(UNIT=5,FILE='FLP5.DAT')
      READ (5,*)D,CO,CT,EPS
      CLOSE(UNIT=5)
C     D=ROTOR DIAMETER,FT.
C     CO=ROOT CHORD AT R=0, FT.
C     CT=TIP CHORD, FT.
C     EPS=DIMENSIONLESS HINGE OFFSET
      OPEN(UNIT=6,FILE='FLP6.DAT')
      READ (6,*)KBETA,THETAT,MW,MIF,V
      CLOSE(UNIT=6)
C     KBETA=PITCH-FLAP COUPLING
C     THETAT=TOTAL TWIST FROM ROOT TO TIP IN DEGS, NEGATIVE FOR WASHOUT
C     MW=BLADE WEIGHT MOMENT ABOUT FLAPPING HINGE IN FOOT-POUNDS
C     MIF=BLADE MASS MOMENT OF INERTIA ABOUT FLAPPING AXIS
C     V=FORWARD VELOCITY IN KTS
      OPEN(UNIT=7,FILE='FLP7.DAT')
      READ (7,*)VT,THETA0,THETA1,THETA2,ALPHA,A1,B1,BETA0
      CLOSE(UNIT=7)
C     VT=TIP SPEED DUE TO ROTATION IN FPS
C     THETA0=INITIAL TRIM COLLECTIVE PITCH, DEGS
C     THETA1=INITIAL TRIM LATERAL CYCLIC PITCH, DEGS.
C     THETA2=INITIAL TRIM LONGITUDINAL CYCLIC PITCH, DEGS.
C     ALPHA=INITIAL TRIM DISC PLANE ANGLE OF ATTACK, DEGS.
C     A1=LONGITUDINAL FLAPPING IN DEGREES
C     B1=LATERAL FLAPPING IN DEGREES
C     BETA0=CONING ANGLE IN DEGREES
      OPEN(UNIT=8,FILE='FLP8.DAT')
      READ (8,*)DELX,DELPSI,RHO,TAVG
      CLOSE(UNIT=8)
      WRITE(*,509)
      WRITE(*,503)CASE,BETLIM
      WRITE(*,504)DELFB1,DELFB2
      WRITE(*,505)FB1LIM,FB2LIM
      WRITE(*,506)CRATE1,CRATE2
      WRITE(*,507)THE1LM,THE2LM
      WRITE(*,508)TOFF,N,NPRED
503   FORMAT(' ','CASE=',F9.3,'     BETLIM=',F9.3)
504   FORMAT(' ','DELFB1=',F9.3,'     DELFB2=',F9.3)
```

```
505   FORMAT(' ','FB1LIM=',F9.3,'     FB2LIM=',F9.3)
506   FORMAT(' ','CRATE1=',F9.3,'     CRATE2=',F9.3)
507   FORMAT(' ','THE1LM=',F9.3,'     THE2LM=',F9.3)
508   FORMAT(' ','TOFF=',F9.3,'   N=',F9.3,'    NPRED=',F9.3)
509   .ORMAT('1')
      WRITE(*,*)'PROGRAM IS HALTED FOR OPPORTUNITY TO PRINT SCREEN'
      WRITE(*,*)'INPUT 1 (OR ANYTHING) TO CONTINUE'
      READ(*,*)DUMMY
      OPEN(UNIT=20,FILE='FLAP.DAT',STATUS='NEW')
      WRITE(20,501)CASE,BETLIM,DELFB1,DELFB2,FB1LIM,FB2LIM
      WRITE(20,502)N,NPRED,FPRED,CRATE1,CRATE2,THE1LM,THE2LM,TOFF
501   FORMAT(6F9.3)
502   FORMAT(8F9.3)
C     DELX=INCREMENT IN DIMENSIONLESS BLADE RADIUS FOR NUMERICAL
C          INTEGRATION OF THRUST WITH RADIUS
C     DELPSI=INCREMENT IN AZIMUTH ANGLE, DEGS, FOR NUMERICAL
C            INTEGRATION OF BLADE MOTION WITH PSI
C     RHO=AIR MASS DENSITY, SLUGS/CU. FT.
C     TAVG=INITIAL AVERAGE THRUST OF ROTOR IN ONE REVOLUTION
      AREA=PI*D*D/4.
      V=V*1.69
      BETLIM=BETLIM*DTR
      DELFB1=DELFB1*DTR
      DELFB2=DELFB2*DTR
      FB1LIM=FB1LIM*DTR
      FB2LIM=FB2LIM*DTR
      THETA0=THETA0*DTR
      THETA1=THETA1*DTR
      THETA2=THETA2*DTR
      THETAT=THETAT*DTR
      A1=A1*DTR
      B1=B1*DTR
      BETA0=BETA0*DTR
      THE1I=THETA1
      THE2I=THETA2
      CRATE1=CRATE1*DTR
      CRATE2=CRATE2*DTR
      THE1LM=THE1LM*DTR
      THE2LM=THE2LM*DTR
      ALPHA=ALPHA*DTR
      OMEGA=VT/D*2
      DELPSI=DELPSI*DTR
      DELT=DELPSI/OMEGA
      DELTPR=DELT*PRT
      PSIMAX=NMAX*TWOPI
      W=0.0
      DELR=DELX*D/2
      TPRED=FPRED*TWOPI/OMEGA
C     INITIALIZATION BEGINS
      TPRINT=0.0
      TIME=0.
      PSI=0.0
      PSII=0.0
      BETA=BETA0-A1
      BETAD=-OMEGA*B1
      SWTCH2=0.0
      FB=0.0
      DELT1=0.0
      KNTRL=1
      TCALC=0.0
      KNTRL=1
      PSILIM=N*TWOPI
      TCONI=PSILIM/OMEGA
      CALL DWNWSH(V,TAVG,RHO,AREA,A1,ALPHA,W)
1     CONTINUE
      IF(KNTRL.EQ.1)GO TO 2
10    CONTINUE
      TCON=TIME-TCONI
      IF(PSI.GT.PSIMAX)GO TO 5000
      CALL SUBCNTRL
2     CONTINUE
      CALL SUBFLAP
3     CONTINUE
      DELT2=T
      TCALC=TCALC+(DELT1+DELT2)/2*DELPSI
      DELT1=DELT2
      TIME=TIME+DELT
      PSI=PSI+DELPSI
4     CONTINUE
      IF(KNTRL.EQ.2)GO TO 8
      IF(KNTRL.EQ.3)GO TO 11
5     CONTINUE
      IF(PSI.GE.PSII+TWOPI)GO TO 19
      IF(PSI.GE.PSILIM)GO TO 18
6     CONTINUE
      IF(KNTRL.EQ.2)GO TO 7
      IF(KNTRL.EQ.3)GO TO 7
      IF(TIME.LT.TPRINT)GO TO 7
      TH1DG=THETA1/DTR
      TH2DG=THETA2/DTR
      REV=PSI/TWOPI
      BETADG=BETA/DTR
      FDBK1G=FDBK1/DTR
      FDBK2G=FDBK2/DTR
      WRITE(*,*)'TIME=',TIME,'   REV=',REV, '   BETA=',BETADG
      WRITE(20,500)TIME,TH1DG,TH2DG,FDBK1G,FDBK2G,REV,BETADG
500   FORMAT(7F8.2)
      IF(PSI.GE.PSIMAX)GO TO 5000
      TPRINT=TPRINT+DELTPR
7     CONTINUE
      IF(KNTRL.EQ.1)GO TO 2
      GO TO 10
19    PSII=PSII+TWOPI
      TAVG=TCALC/TWOPI
      CALL DWNWSH(V,TAVG,RHO,AREA,A1,ALPHA,W)
      TCALC=0.0
      IF(PSI.GE.PSILIM)GO TO 18
      GO TO 15
```

```
18    CONTINUE
      WRITE(*,*)'PSILIM=',PSILIM,'    KNTRL=',KNTRL
      IF(KNTRL.EQ.2)THEN
         WRITE(*,*)'KNTRL=',KNTRL
         PSII=PSIII
         PSI=PSII
         BETA=BETAI
         BETAD=BETADI
         W=WI
         TIME=TIMEI
         TC=TPRED
         FB=FB-SWTCH2
         GO TO 21
      ENDIF
      KNTRL=2
      IF(PSI.GE.PSIMAX)GO TO 5000
      WRITE(*,*)'KNTRL=2'
20    CONTINUE
      PSII=PSI
      PSIPRI=PSIPRT
      PSILIM=PSI+NPRED*TWOPI
      TIMEI=TIME
      BETAI=BETA
      BETADI=BETAD
      WI=W
      PSIII=PSI
      GO TO 1
8     IF(ABS(BETA).GT.BETLIM)GO TO 9
      SWTCH2=1.0
      GO TO 5
9     DELBT2=ABS(BETA)-BETLIM
      IF(DELBT1.GE.DELBT2)GO TO 14
      DELBT1=DELBT2
      GO TO 5
15    CONTINUE
      DELT1=0.
      TCALC=0.
      GO TO 6
14    CONTINUE
      SWTCH2=0.0
      FB=FB+1.0
      SWITCH=1.0
      IF(COS(PSI).GE.SIN(PSI))SWITCH=2.0
      WRITE(*,*)'KNTRL=3'
      TC=TPRED*(PSI-PSIII)/TWOPI/NPRED
21    CONTINUE
      KNTRL=3
      PSIPRT=PSIPRI
      PSI=PSIII
      TIME=TIMEI
      BETA=BETAI
      BETAD=BETADI
      W=WI
      GO TO 15
```

```
11      CONTINUE
        IF(TIME.LT.TPRINT)GO TO 12
        TH1DG=THETA1/DTR
        TH2DG=THETA2/DTR
        REV=PSI/TWOPI
        BETADG=BETA/DTR
        FDBK1G=FDBK1/DTR
        FDBK2G=FDBK2/DTR
        WRITE(*,*)'TIME=',TIME,'   REV=',REV, '   BETA=',BETADG
        WRITE(20,500)TIME,TH1DG,TH2DG,FDBK1G,FDBK2G,REV,BETADG
        IF(PSI.GE.PSIMAX)GO TO 5000
        TPRINT=TPRINT+DELTPR
12      CONTINUE
        IF(TIME.GE.TIMEI+TC)GO TO 13
        GO TO 10
13      CONTINUE
        KNTRL=2
        DELBT1=0.
        GO TO 20
5000    CLOSE(UNIT=20)
        STOP
        END

        SUBROUTINE SUBCNTRL
        COMMON/COM1/THE1LM,THE2LM,KNTRL,TCON,TOFF,THE1I,THE2I,SWITCH
        COMMON/COM2/DELFB1,DELFB2,FB,PI,CRATE1,CRATE2,FDBK1,FDBK2
        COMMON/COM5/PSI,EPS,CO,CT,THETA0,THETAT,THETA1,THETA2,KBETA,BETA
        COMMON/COM6/FB1LIM,FB2LIM
        REAL KBETA
1       IF(TCON.GT.TOFF)GO TO 2
        DELTH1=CRATE1*TCON
        DELTH2=CRATE2*TCON
        IF(DELTH1.GT.THE1LM)DELTH1=THE1LM
        IF(DELTH2.GT.THE2LM)DELTH2=THE2LM
        THETA1=THE1I+DELTH1
        THETA2=THE2I+DELTH2
        GO TO 3
2       DELTH1=CRATE1*(TCON-TOFF)
        DELTH2=CRATE2*(TCON-TOFF)
        IF(DELTH1.GT.THE1LM)DELTH1=THE1LM
        IF(DELTH2.GT.THE2LM)DELTH2=THE2LM
        THETA1=THE1I+THE1LM-DELTH1
        THETA2=THE2I+THE2LM-DELTH2
3       IF(FB.LE.0.)FB=0.0
        FDBK1=FB*DELFB1
        FDBK2=FB*DELFB2
        IF(FDBK1.GT.FB1LIM)FDBK1=FB1LIM
        IF(FDBK2.GT.FB1LIM)FDBK2=FB2LIM
        IF(SWITCH.EQ.1.)FDBK2=0.0
        IF(SWITCH.EQ.2.)FDBK1=0.0
        THETA1=THETA1-FDBK1
        THETA2=THETA2-FDBK2
        IF(THETA1.LT.THE1I)THETA1=THE1I
        IF(THETA2.LT.THE2I)THETA2=THE2I
```

```
4       RETURN
        END

        SUBROUTINE DWNWSH(V,TAVG,RHO,AREA,A1,ALPHA,W)
        W=SQRT(0.5*(-V**2+SQRT(V**4+(TAVG/RHO/AREA)**2)))
        DO 1 I=1,10
        VPRIME=SQRT((V-W*SIN(ALPHA+A1))**2+(W*COS(ALPHA+A1))**2)
        W=TAVG/2/RHO/VPRIME/AREA
1       CONTINUE
        RETURN
        END
        SUBROUTINE SUBFLAP
        REAL KBETA,MW,MIF,M1,M2,MAERO
        COMMON/COM3/CLI(20),ALPHAL(20),CDI(33),ALPHAD(33),ALPHAB,CL,CD
        COMMON/COM4/OMEGA,V,ALPHA,W,BETAD,RHO,DELPSI,DELR,T,MIF,MW,D
        COMMON/COM5/PSI,EPS,CO,CT,THETA0,THETAT,THETA1,THETA2,KBETA,BETA
        PI=3.14159
        DELT=DELPSI/OMEGA
        SALPHA=SIN(ALPHA)
        CALPHA=COS(ALPHA)
C       REM START OF INTEGRATION OF THRUST OVER RADIUS AT A PARTICULAR PSI
1       T=0
        SPSI=SIN(PSI)
        CPSI=COS(PSI)
        EPSR=EPS*D/2
        R=EPSR
        M1=0
        DT1=0
        MAERO=0
C       REM RETURN TO HERE AFTER INCREMENTING RADIUS, R
2       X=R/D*2
        C=CO-(CO-CT)*X
C       THETA=BLADE PITCH ANGLE
        THETA=THETA0+THETAT*X+THETA1*CPSI+THETA2*SPSI+KBETA*BETA
C       VTHETA=RESULTANT TANGENTIAL VELOCITY
        VTHETA=OMEGA*R+V*CALPHA*SPSI
C       VU=RESULTANT VELOCITY UP THROUGH DISK
        VU=V*SALPHA-W-(R-EPSR)*BETAD-BETA*V*CALPHA*CPSI
        ALPHAB=ATAN(ABS(VU/VTHETA))
C       START OF LOGIC TO DETERMINE CL AND CD AND RESOLUTION FACTORS
        CLFAC=1.0
        CDFAC=1.0
        IF (VTHETA.GE.0.0.AND.VU.GE.0.0)THEN
             ALPHAB=THETA+ALPHAB
             IF(ALPHAB.LT.0.0)CLFAC=-1.
             ALPHAB=ABS(ALPHAB)
             GO TO 3
        ENDIF
        IF (VTHETA.GT.0.0.AND.VU.LT.0.0) THEN
              ALPHAB=THETA-ALPHAB
              IF(ALPHAB.LT.0.0)CLFAC=-1.
              ALPHAB=ABS(ALPHAB)
              GO TO 3
        ENDIF
```

```
      IF (VTHETA.LT.0.0.AND.VU.LT.0.0) THEN
            ALPHAB=PI-ALPHAB-THETA
            CLFAC=-1.0
            CDFAC=-1.0
            IF(ALPHAB.GT.PI)CLFAC=1.0
            IF(ALPHAB.GT.PI)ALPHAB=2.*PI-ALPHAB
            GO TO 3
      ENDIF
      IF (VTHETA.LT.0.0.AND.VU.GT.0.0) THEN
             ALPHAB=PI+THETA-ALPHAB
             CLFAC=1.0
             CDFAC=-1.0
             IF(ALPHAB.GT.PI)CLFAC=-1.0
             IF(ALPHAB.GT.PI)ALPHAB=2.*PI-ALPHAB
      ENDIF
C     END OF LOGIC ON CL, CD AND RESOLUTION FACTORS
3     CALL AIRFOIL
      CL=CL*CLFAC
      CD=CD*CDFAC
      VRSQD=VU**2+VTHETA**2
      DLDR=RHO/2*VRSQD*CL*C
      DDDR=RHO/2*VRSQD*CD*C
C     DLDR AND DDDR ARE LIFT AND DRAG DERIVATIVES WRT TO R
      VR=SQRT(VRSQD)
      DT2=DLDR*ABS(VTHETA/VR)+DDDR*ABS(VU/VR)
      M2=(R-EPSR)*DT2
      DMDR=(M1+M2)/2
      DTDR=(DT1+DT2)/2
C     DTDR AND DMDR ARE RADIAL THRUST AND MOMENT DERIVATIVES WRT R
      M1=M2
      DT1=DT2
      MAERO=MAERO+DMDR*DELR
      T=T+DTDR*DELR
C     T AND MAERO ARE BLADE THRUST AND MOMENT AT INSTANT OF TIME
      R=R+DELR
      IF (R.GT.D/2) GO TO 4
      GOTO 2
4     BETADD=MAERO/MIF-BETA*OMEGA**2-MW/MIF
      BETAD=BETAD+BETADD*DELT
      BETA=BETA+BETAD*DELT+BETADD*DELT**2/2.
      M1=0.
      DT1=0.
      PSI2=PSI
      PSI1=PSI2
C     T IS THE BLADE THRUST AT AN INSTANT OF TIME AND PSI
      RETURN
      END

C     SUBROUTINE FOR CL AND CD AS FUNCTION OF ALPHA
      SUBROUTINE AIRFOIL
      COMMON/COM3/CLI(20),ALPHAL(20),CDI(33),ALPHAD(33),ALPHAB,CL,CD
      DO 1 I=2,20
      IF(ALPHAB.EQ.ALPHAL(I)) THEN
            CL=CLI(I)
            GO TO 2
      ENDIF
      IF(ALPHAB.LE.ALPHAL(I)) THEN
              AI=CLI(I)
              BI=CLI(I-1)
              CI=ALPHAL(I)
              DI=ALPHAL(I-1)
              GO TO 5
      ENDIF
1     CONTINUE
5     CL=(AI-BI)*(ALPHAB-DI)/(CI-DI)+BI
2     CONTINUE
      DO 3 I=2,33
      IF(ABS(ALPHAB).EQ.ALPHAD(I)) THEN
           CD=CDI(I)
           GO TO 4
      ENDIF
      IF(ABS(ALPHAB).LE.ALPHAD(I)) THEN
              AI=CDI(I)
              BI=CDI(I-1)
              CI=ALPHAD(I)
              DI=ALPHAD(I-1)
              GO TO 6
      ENDIF
3     CONTINUE
6     CD=(AI-BI)*(ALPHAB-DI)/(CI-DI)+BI
4     RETURN
      END
```

FORTRAN listing for PROGRAM DCFLAP
(digital controller)

```
      PROGRAM DCFLAP
C-------------------------------------------------------------------
C      Program to simulate the helicopter rotor blade flapping
C      motion with real-time control system preventing excessive
C      flapping. The control system can be cut off by setting
C      GAIN=0.
C      Flapping angle BETA and flapping angular velocity BETAD
C      as a function of NREV (number of revolutions) are stored
C      in file RESULT.DAT.
C      Pitch angles THETA0 (collective), THETA1 (lateral cyclic),
C      and THETA2 (longitudinal cyclic) as a function of NREV are
C      stored in file CONTROL.DAT.
C      Feedback gain GAIN of the control system as a function of
C      NREV is stored in GAIN.DAT. (An adaptive gain technique
C      is used.)
C
C      Aerospace Engineering Department
C      Pennsylvania State University
C      University Park,  PA 16802
C-------------------------------------------------------------------
C
C
      CHARACTER REPLY*1,TRIM*5
      REAL KBETA,MW,MIF,M1,M2,MAERO,NMAX,NBET,MU,
     &     LAMDA,NREV,NPRED
      COMMON/PMT/PI,DTR,STEP,NMAX,PSIB,DPSIB,W
      COMMON/INPUT/THET0,THET1,THET2,ALPHA,BET1,BET2
      COMMON/SIGNAL/TRIM,VAR1,VAR2,VAR3,VAR4
      COMMON/PTB/PERTB,D0,D1,D2,DALPHA
      COMMON/SIMU/B,D,C0,CT,EPS,KBETA,THETAT,MW,MIF,V,VT,DELX
     &,RHO,KTEN,CRATE0,CRATE1,CRATE2,CRATEA,GAIN,SOS,BEEP,U,U0,NPRED
      DATA TRIM/'EXIST'/
C
C------------------------------------------------------------------------
C  FILE 'RESULT.DAT' STORES THE VALUES OF PSIB,BETA,BETA DOT (IN DEGREES)
C  FILE 'TRIM.DAT' STORES THE CONTROL INPUTS FOR THE CURRENT TRIM
C  FILE 'IC.DAT' STORES THE INITIAL CONDITION TABLE
C
       OPEN(100,FILE='RESULT.DAT',STATUS='NEW')
       OPEN(101,FILE='TRIM.DAT',STATUS='UNKNOWN')
       OPEN(102,FILE='IC.DAT',STATUS='UNKNOWN')
       OPEN(107,FILE='GAIN.DAT',STATUS='UNKNOWN')
               REWIND 3
               REWIND 101
               REWIND 102
C --------------------------
C   NBET --- THE NUMBER OF POSITIONS PER REVOLUTION TO BE CALCULATED
          NBET=72
C
C STEP -- STEP IN THE INITIAL CONDITION TABLE
        STEP=360./NBET
C --------------------------
       PI = 3.14159265
       DTR=PI/180.
```

```
C ------------------------------------------------------------------
C HAVE YOU GENERATED THE TRIM (OR NORMAL) FLIGHT CONDITIONS?
C NO-->GOTO 1010 TO INPUT THE INITIAL VALUES NEEDED TO GENERATE:
C      'TRIM.DAT' & 'IC.DAT'
C YES-->READ THE INITIAL VALUES FROM THE TWO FILES 'TRIM.DAT' & 'IC.DAT'
C      THEN, GOTO 1020
C
1000  WRITE(6,*) 'Have you generated the trim condition?(Y/N):'
      READ(5,31) REPLY
      IF(REPLY.EQ.'N'.OR.REPLY.EQ.'n') THEN
      TRIM='NONE'
      GOTO 1010
      ELSE IF(REPLY.EQ.'Y'.OR.REPLY.EQ.'y') THEN
      READ(101,32) THETOD,THET1D,THET2D,ALPHAD,W
32    FORMAT(1X,5G15.8)
      READ(102,35) PSIBD,BET10D,BET20D
           REWIND 101
           REWIND 102
      GOTO 1020
      ELSE
      GOTO 1000
      ENDIF
1010  WRITE(6,*) 'Enter one blade azimuth angle in degrees:'
      READ(5,*) PSIBD
      WRITE(6,*) 'Enter collective in degrees:'
      READ(5,*) THETOD
      WRITE(6,*) 'Enter longitudinal cyclic in degrees:'
      READ(5,*) THET2D
      WRITE(6,*) 'Enter lateral cyclic in degrees:'
      READ(5,*) THET1D
      WRITE(6,*) 'Enter ALPHA in degrees:'
      READ(5,*) ALPHAD
      BET10D=0.
      BET20D=0.
      W=0.0
        GOTO 1300
C ------------------------------------------------------------------
C THE FOLLOWING BLOCK ALLOWS YOU TO EXAMINE THE EFFECTS OF EACH CONTROL
C INPUTS OR DISTURBANCES ON THE ROTOR FLAPPING
C
1020  WRITE(6,*) 'These are the current inputs:'
      WRITE(6,34) THETOD,THET2D,THET1D,ALPHAD
34    FORMAT(1X,'Collective is',G15.8,'degrees'
     &       /1X,'Longitudinal cyclic is',G15.8,'degrees'
     &       /1X,'Lateral cyclic is',G15.8,'degrees'
     &       /1X,'ALPHA is',G15.8,'degrees')
      IF(TRIM.EQ.'NONE') THEN
      PERTB=1.E+30
      GOTO 1205
      ELSE
1100  WRITE(6,*) 'Do you want to give perturbations?(Y/N):'
      READ(5,31) REPLY
      IF(REPLY.EQ.'Y'.OR.REPLY.EQ.'y') THEN
      GOTO 1101
      ELSE IF(REPLY.EQ.'N'.OR.REPLY.EQ.'n') THEN
      GOTO 1300
      ELSE
      GOTO 1100
      ENDIF
1101  WRITE(6,*) 'Enter number of revolutions at which the ',
     &'perturbations are given:'
      READ(5,*) PERTB
1200  WRITE(6,*) 'Enter perturbations on Collective, Longitudinal',
     &'Cyclic, Lateral Cyclic, and ALPHA in degrees:'
      READ(5,*) D0,D1,D2,DALPHA
      D0=D0*DTR
      D1=D1*DTR
      D2=D2*DTR
      DALPHA=DALPHA*DTR
31    FORMAT(A1)
C---------------------------------------------------------------------
      WRITE(6,*) 'Enter changing rates on Collective, Longitudinal',
     &'Cyclic, Lateral Cyclic, and ALPHA in deg/sec:'
      READ(5,*) CRATE0,CRATE1,CRATE2,CRATEA
C Transform CRATE0,CRATE1,CRATE2,CRATEA to RAD/SEC
      CRATE0=CRATE0*DTR
      CRATE1=CRATE1*DTR
      CRATE2=CRATE2*DTR
      CRATEA=CRATEA*DTR
C
      WRITE(6,*) 'INPUT FEEDBACK GAIN:'
      READ(5,*) GAIN
      ENDIF
1205  CONTINUE
C---------------------------------------------------------------------
C
C ====================================================================
C  THE FOLLOWING BLOCK PERFORMS LINEAR INTERPOTATION IN THE INITIAL CONDITION
C  TABLE TO GET THE CORRECT INITIAL CONDITION FOR THE PSIB INPUT
C
C NENTRY-- THE ENTRY PLACE NUMBER-1 IN THE INITIAL CONDITION TABLE
1206      NENTRY=JNINT(PSIBD/STEP-.5)
          DO 1310  JUMP=1,NENTRY
1310      READ(102,35) VNUL
          READ(102,35) PSIBD1,BET1D1,BET2D1
          READ(102,35) PSIBD2,BET1D2,BET2D2
          BET10D=BET1D1+(PSIBD-PSIBD1)/(PSIBD2-PSIBD1)*(BET1D2-BET1D1)
          BET20D=BET2D1+(PSIBD-PSIBD1)/(PSIBD2-PSIBD1)*(BET2D2-BET2D1)
35      FORMAT(1X,3G15.8)
        REWIND 102
C
C
C====================================================================
1300         PSIB=PSIBD*DTR
             THET0=THET0D*DTR
           THET1=THET1D*DTR
           THET2=THET2D*DTR
           BET10=BET10D*DTR
```

```
            BET20-BET20D*DTR
              ALPHA-ALPHAD*DTR
            NMAX-10.
            DPSIB-2.*PI/NBET
        IF(TRIM.EQ.'EXIST') THEN
      NREV-PSIB/2./PI
        WRITE(100,35) NREV,BET10D,BET20D
        ENDIF
C
        CALL BWFLAP
        REWIND 102
C
C THE FOLLOWING WRITE STATEMENT RECORDS THE CONTROL INPUTS FOR THE CURRENT
C    TRIM CONDITION
        IF(TRIM.EQ.'NONE') THEN
        WRITE(101,32) THETOD,THET1D,THET2D,ALPHAD,W
        REWIND 101
        WRITE(6,36)
36    FORMAT(30(/),10X,'TRIM GENERATED!',5(/))
        TRIM-'EXIST'
        GOTO 1020
        ENDIF
            STOP
            END

C------------------------------------------------------------------------------
      SUBROUTINE WRITE
        REAL NMAX,NREV
        CHARACTER TRIM*5
        COMMON/PMT/PI,DTR,STEP,NMAX,PSI,DELPSI,W
        COMMON/INPUT/THETA0,THETA1,THETA2,ALPHA,BETA,BETAD
        COMMON/SIGNAL/TRIM,NREV,VAR1,VAR2,VAR3
        PSID-PSI/DTR
        BET-BETA/DTR
        BETD-BETAD/DTR
      IF(TRIM.EQ.'EXIST') THEN
        WRITE(100,35) NREV,BET,BETD
35      FORMAT(1X,3G15.8)
        ELSE IF(PSI.GE.(16.*PI-STEP*PI/360.).AND.PSI.LE.(18.*PI)) THEN
        PSI1ST-(PSI-16.*PI)*180./PI
        WRITE(102,35) PSI1ST,BET,BETD
        ENDIF
        RETURN
        END


C
      SUBROUTINE BWFLAP
      CHARACTER BEEP*1,DETECT*1,TRIM*5
      REAL UTHET1(2),UTHET2(2)
      REAL KBETA,MW,MIF,M1,M2,MAERO,NMAX,NBET,MU,
     &     LAMDA,NREV,NPRED
      DIMENSION U(10),U0(10)
      COMMON/PMT/PI,DTR,STEP,NMAX,PSI,DELPSI,W
      COMMON/INPUT/THETA0,THETA1,THETA2,ALPHA,BETA,BETAD
      COMMON  CLI(20),ALPHL(20),CDI(33),ALPHD(33),ALPHAB,CL,CD
      COMMON/SIMU/B,D,C0,CT,EPS,KBETA,THETAT,MW,MIF,V,VT,DELX
     &,RHO,KTEN,CRATE0,CRATE1,CRATE2,CRATEA,GAIN,SOS,BEEP,U,U0,NPRED
      COMMON/PTB/PERTB,D0,D1,D2,DALPHA
      COMMON/HOLD/UTHET1,UTHET2
      COMMON/SIGNAL/TRIM,NREV,A1,B1,AREA
      COMMON/CLOSED/T1,T2,T3,T4,F1,F2,F3,F4,DENOM,A11,A12,A13,A14,
     & B11,TAU,CBAR,GAMMAF,SIGMA,ASIG
C NPRED is number of predict-ahead revolutions
      NPRED=3.
C
      DETECT='0'
      OPEN(UNIT=1,FILE='CL.DAT',STATUS='OLD')
      DO 1 I=1,20
      READ(1,*) ALPHL(I),CLI(I)
1     CONTINUE
      CLOSE (UNIT=1)
      OPEN(UNIT=2,FILE='CD.DAT',STATUS='OLD')
      DO 3 I=1,33
      READ(2,*) ALPHD(I),CDI(I)
3     CONTINUE
      CLOSE(UNIT=2)
      PI=3.14159
      DO 5 I=1,20
      ALPHL(I)=ALPHL(I)*DTR
5     CONTINUE
      DO 6 I=1,33
      ALPHD(I)=ALPHD(I)*DTR
6     CONTINUE
      PSII=0.0
C     PROGRAM READS IN ORDER:        FLP5.DAT
C                               1. NUMBER OF BLADES
C                               2. ROTOR DIAMETER
C                               3. ROOT CHORD AT R=0
C                               4. TIP CHORD
C                               5. DIMENSIONLESS HINGE OFFSET
C
C                                    FLP6.DAT
C                               1. PITCH-FLAP COUPLING, KBETA
C                               2. TOTAL TWIST, DEGS.,NEGATIVE FOR WASHOUT
C                               3. BLADE WEIGHT MOMENT ABOUT FLAPPING AXIS
C                               4. MASS MOMENT OF INERTIA ABOUT FLAPPING AXIS
C                               5. TIP SPEED IN FPS
C
C                                     FLP7.DAT
C                               1. DIMENSIONLESS INCREMENT IN RADIUS, DELX
C                               2. INCREMENT IN AZIMUTH ANGLE,DEGS, DELPSI
C                               3. NUMBER OF DELPSI BETWEEN PRINTOUTS
C
C                                      FLP8.DAT
C                               1. FORWARD SPEED IN KTS.
C                               2. AIR DENSITY, SLUGS/CU.FT.
C                               3. COLLECTIVE PITCH, THETA0,DEGREES
```

```
C                                  4. LATERAL PITCH,THETA1, DEGREES
C                                  5. LONGITUDINAL PITCH,THETA2.DEGREES
C                                  6. DISC PLANE ANGLE OF ATTACK,ALPHA,DEGS.
C
C         FLP5.DAT    2,44,2.25,2.25,.01
C         FLP6.DAT    0,-10,3122,1422,738
C         FLP7.DAT   .05,5,5
C         FLP8.DAT    80,.002378,15.1,1.9,-1.66,-4.63 (FWD FLIGHT)
C         FLP8.DAT    0,.002378,15,0,0,0              (HOVER)
C
      OPEN(UNIT=5,FILE='FLP5.DAT',STATUS='OLD')
      READ (5,*)B,D,CO,CT,EPS
      CLOSE(UNIT=5)
      OPEN(UNIT=6,FILE='FLP6.DAT',STATUS='OLD')
      READ (6,*)KBETA,THETAT,MW,MIF,VT
      CLOSE(UNIT=6)
      OPEN(UNIT=7,FILE='FLP7.DAT',STATUS='OLD')
      READ (7,*) DELX
      CLOSE(UNIT=7)
      OPEN(UNIT=8,FILE='FLP8.DAT',STATUS='OLD')
      READ (8,*) V,RHO
      CLOSE(UNIT=8)
      OPEN(105,FILE='CONTROL.DAT',STATUS='UNKNOWN')
      V=V*1.69
      AREA=PI*D*D/4.
      OMEGA=VT/D*2.
      DELT=DELPSI/OMEGA
      DELR=DELX*D/2.
      PSI0=PSI
      TAVG=0.
      TTAVG=0.
C KTEN specifies the sampling period
      KTEN=30.*DTR/DELPSI+0.5
C
      IKTEN=0
      THETAT=THETAT*DTR
      MU=V/VT
      T1=(.941+MU**2/2)/2
      T2=(.941/3.+MU**2/2.)*.97
      T3=.941/4.*(.941+MU**2)
      T4=MU/2.*(.941+MU**2/4.)
      F1=.97**3/3.
      F2=.941/4.*(.941+MU**2)
      F3=.97**3*(.941/5.+MU**2/6.)
      F4=MU*.97**3/3.
      DENOM=.941-MU**2/2.
      A11=4.*(MU*.941/2.-MU**3/8.)/.941/DENOM
      A12=8.*MU*.97/3./DENOM
      A13=2.*MU*.941/DENOM
      A14=(.941+3./2.*MU**2)/DENOM
      B11=4.*MU*.97/3./(.941+MU**2/2.)
      TAU=MW/MIF/OMEGA**2
      CBAR=(CO+CT)/2.
      GAMMAF=CBAR*RHO*5.7*D**4/32./MIF
      SIGMA=B*CBAR/PI/D*2.
      ASIG=5.7*SIGMA
      BETA0=0.0
      B1=0.0
C-------------------------------------------------------------------------
C WRITE INITIAL CYCLIC INPUTS AND ZERO CONTROL INPUTS TO FILE "CONTROL.DAT"
C
      NREV=PSI/2./PI
      CNTRL1=0.
      CNTRL2=0.
      E1=THETA1/DTR
      E2=THETA2/DTR
      WRITE(105,61) NREV,CNTRL1,CNTRL2,E1,E2
61    FORMAT(1X,5G15.8)
C-------------------------------------------------------------------------
      THRST1=0.
2280  CONTINUE
C------------------------------------------------------------------
C     START OF INTEGRATION LOOP FOR PSI
C
C     RETURN TO HERE AFTER INCREMENTING PSI
C------------------------------------------------------------------
C  THIS BLOCK STIPULATES THE PILOT INPUTS.
C  CRATE0,CRATE1,CRATE2,CRATEA ARE THE INPUT RATES OF THETA0,THETA1,THETA2,ALPHA
C  RESPECTIVELY
      NREV=PSI/2./PI
      IF(NREV.GE.PERTB) THEN
      IF(DETECT.EQ.'0') THEN
      THETA0=THETA0+D0
      THETA1=THETA1+D1
      THETA2=THETA2+D2
      ALPHA=ALPHA+DALPHA
      DETECT='1'
      ELSE
      THETA0=THETA0+CRATE0*DELPSI/OMEGA
      THETA1=THETA1+CRATE1*DELPSI/OMEGA
      THETA2=THETA2+CRATE2*DELPSI/OMEGA
      ALPHA=ALPHA+CRATEA*DELPSI/OMEGA
      ENDIF
      ENDIF
C
      UTHET1(1)=UTHET1(2)
      UTHET1(2)=THETA1
      UTHET2(1)=UTHET2(2)
      UTHET2(2)=THETA2
C
C---------------------------------------------------------------------
      SPSI=SIN(PSI)
      CPSI=COS(PSI)
      SALPHA=SIN(ALPHA)
      CALPHA=COS(ALPHA)
      LAMDA=MU*ALPHA
      DO 9050 I=1,20
      CTR=ASIG/2.*(LAMDA*T1+(THETA0+KBETA*BETA0)*T2+THETAT*T3+(THETA2
     1-KBETA*B1)*T4)
      CFAC=V**4+(TTAVG/RHO/AREA)**2+W**3*2.*V*SIN(A1+ALPHA)
      IF(CFAC.LT.0.) THEN
         W=SQRT(0.5*(-V**2+SQRT(V**4+(TTAVG/RHO/AREA)**2)))
         GO TO 9051
      ENDIF
      WFAC=-V**2+SQRT(CFAC)
      IF(WFAC.LT.0.)THEN
         W=SQRT(0.5*(-V**2+SQRT(V**4+(TTAVG/RHO/AREA)**2)))
         GO TO 9051
      ENDIF
      W=SQRT(0.5*(-V**2+SQRT(CFAC)))
9051  WVT=W/VT
      LAMDA=MU*ALPHA-WVT
      BETA0=GAMMAF*(LAMDA*F1+(THETA0+KBETA*BETA0)*F2+THETAT*F3+(THETA2-
     1KBETA*B1)*F4)-TAU
      A1=LAMDA*A11+(THETA0+KBETA*BETA0)*A12+THETAT*A13+(THETA2-KBETA*B1
     1)*A14
      B1=BETA0*B11-(THETA1-KBETA*A1)
9050  TTAVG=RHO*AREA*VT**2*CTR
      THRST2=0
C     START OF INTEGRATION OVER X FROM XH TO 1
      EPSR=EPS*D/2
      R=EPSR
      M1=0
      DT1=0
      MAERO=0
C     RETURN TO HERE AFTER INCREMENTING RADIUS, R
2380  X=R/D*2
      C=C0-(C0-CT)*X
      THETA=THETA0+THETAT*X+THETA1*CPSI+THETA2*SPSI+KBETA*BETA
      VTHETA=OMEGA*R+V*CALPHA*SPSI
      VU=V*SALPHA-W-(R-EPSR)*BETAD-BETA*V*CALPHA*CPSI
      ALPHAB=ATAN(ABS(VU/VTHETA))
      CLFAC=1.0
      CDFAC=1.0
      IF (VTHETA.GE.0.0.AND.VU.GE.0.0)THEN
           ALPHAB=THETA+ALPHAB
           IF(ALPHAB.LT.0.0)CLFAC=-1.
           ALPHAB=ABS(ALPHAB)
           GO TO 2381
      ENDIF
      IF (VTHETA.GT.0.0.AND.VU.LT.0.0) THEN
            ALPHAB=THETA-ALPHAB
            IF(ALPHAB.LT.0.0)CLFAC=-1.
            ALPHAB=ABS(ALPHAB)
            GO TO 2381
      ENDIF
      IF (VTHETA.LT.0.0.AND.VU.LT.0.0) THEN
            ALPHAB=PI-ALPHAB-THETA
            CLFAC=-1.0
            CDFAC=-1.0
            IF(ALPHAB.GT.PI)CLFAC=1.0
            IF(ALPHAB.GT.PI)ALPHAB=2.*PI-ALPHAB
```

```
          GO TO 2381
      ENDIF
      IF (VTHETA.LT.0.0.AND.VU.GT.0.0) THEN
             ALPHAB=PI+THETA-ALPHAB
             CLFAC=+1.0
             CDFAC=-1.0
             IF(ALPHAB.GT.PI)CLFAC=-1.0
             IF(ALPHAB.GT.PI)ALPHAB=2.*PI-ALPHAB
      ENDIF
2381  CALL AIRFOIL
      CL=CL*CLFAC
      CD=CD*CDFAC
      VRSQD=VU**2+VTHETA**2
      DLDR=RHO/2*VRSQD*CL*C
      DDDR=RHO/2*VRSQD*CD*C
      VR=SQRT(VRSQD)
      DT2=DLDR*ABS(VTHETA/VR)+DDDR*ABS(VU/VR)
      M2=(R-EPSR)*DT2
      DMDR=(M1+M2)/2
      DTDR=(DT1+DT2)/2
      M1=M2
      DT1=DT2
      MAERO=MAERO+DMDR*DELR
      THRST2=THRST2+DTDR*DELR
      R=R+DELR
      IF (R.GT.D/2) GO TO 2690
      GOTO 2380
2690  BETDD2=MAERO/MIF-BETA*OMEGA**2-MW/MIF
      BETAD2=BETAD2+(BETDD1+BETDD2)/2.*DELT
      BETAD=BETAD2
      BETA=BETA+(BETAD1+BETAD2)/2.*DELT
      BETDD1=BETDD2
      BETAD1=BETAD2
      M1=0.
      DT1=0.
      PSI2=PSI
      PSI1=PSI2
      THRUST=(THRST1+THRST2)/2.
      THRST1=THRST2
      TAVG=TAVG+THRUST*DELPSI
C
      CALL WRITE
C
      PSI=PSI+DELPSI
      PSII=PSII+ DELPSI
      IF (PSII.LE.2.*PI) GO TO 9801
      TAVG=TAVG/2./PI*B
      CTRW=TAVG/RHO/AREA/VT**2
      DO 9800 I=1,5
      CFAC=V**4+(TAVG/RHO/AREA)**2+W**3*2.*V*SIN(A1+ALPHA)
      IF(CFAC.LT.0.)THEN
         W=SQRT(0.5*(-V**2+SQRT(V**4+(TAVG/RHO/AREA)**2)))
         GO TO 9800
      ENDIF
      WFAC=-V**2+SQRT(CFAC)
      IF(WFAC.LT.0.)THEN
         W=SQRT(0.5*(-V**2+SQRT(V**4+(TAVG/RHO/AREA)**2)))
         GO TO 9800
      ENDIF
      W=SQRT(0.5*(-V**2+SQRT(CFAC)))
9800  WVT=W/VT
      TAVG=0.
      PSII=0.
9801  IF(PSI.GT.NMAX*2.*PI) GOTO 100
      IKTEN=IKTEN+1
      IF(IKTEN.EQ.KTEN) THEN
C
      IF(TRIM.EQ.'EXIST'.AND.ABS(GAIN).GT.1.E-10) CALL CNTRL
C
      IKTEN=0
      ENDIF
      GOTO 2280
100   RETURN
      END
C
C
C     SUBROUTINE FOR CL AND CD AS FUNCTION OF ALPHA
      SUBROUTINE AIRFOIL
      COMMON CLI(20),ALPHL(20),CDI(33),ALPHD(33),ALPHAB,CL,CD
      DO 3040 I=2,20
      IF(ALPHAB.EQ.ALPHL(I)) THEN
            CL=CLI(I)
            GO TO 4000
      ENDIF
      IF(ABS(ALPHAB).LE.ALPHL(I)) THEN
              AI=CLI(I)
              BI=CLI(I-1)
              CI=ALPHL(I)
              DI=ALPHL(I-1)
              GO TO 3050
      ENDIF
3040  CONTINUE
3050  CL=(AI-BI)*(ALPHAB-DI)/(CI-DI)+BI
      CL=CL*ABS(ALPHAB)/ALPHAB
4000  CONTINUE
      DO 3100 I=2,33
      IF(ABS(ALPHAB).EQ.ALPHD(I)) THEN
           CD=CDI(I)
           GO TO 3200
      ENDIF
      IF(ABS(ALPHAB).LE.ALPHD(I)) THEN
              AI=CDI(I)
              BI=CDI(I-1)
              CI=ALPHD(I)
              DI=ALPHD(I-1)
              GO TO 3110
      ENDIF
3100  CONTINUE
```

```
3110  CD=(AI-BI)*(ALPHAB-DI)/(CI-DI)+BI
3200  RETURN
      END
C
C
      SUBROUTINE CNTRL
      CHARACTER BEEP*1,TRIM*5
      REAL U(10),U0(10),UTHET1(2),UTHET2(2)
      REAL KBETA,MW,MIF,M1,M2,MAERO,NMAX,NBET,
     &     LAMDA,NREV,NPRED
      COMMON/PMT/PI,DTR,STEP,NMAX,PSI0,DELPSI,W0
      COMMON/INPUT/THETA0,THETA1,THETA2,ALPHA,XBETA0,XBETAD0
      COMMON/SIMU/B,D,C0,CT,EPS,KBETA,THETAT,MW,MIF,V,VT,DELX
     &,RHO,KTEN,CRATE0,CRATE1,CRATE2,CRATEA,GAIN,SOS,BEEP,U,U0,NPRED
      COMMON  CLI(20),ALPHL(20),CDI(33),ALPHD(33),ALPHAB,CL,CD
      COMMON/SIGNAL/TRIM,NREV,XA1,XB1,AREA
      COMMON/HOLD/UTHET1,UTHET2
      COMMON/CLOSED/T1,T2,T3,T4,F1,F2,F3,F4,DENOM,A11,A12,A13,A14,
     & B11,TAU,CBAR,GAMMAF,SIGMA,ASIG
      A1=XA1
      B1=XB1
      BETA0=XBETA0
      BETAD0=XBETAD0
      E1=THETA1/DTR
      E2=THETA2/DTR
      W=W0
      BETA=BETA0
      BETAD=BETAD0
      SALPHA=SIN(ALPHA)
      CALPHA=COS(ALPHA)
      OMEGA=VT/D*2
      DELT=DELPSI/OMEGA
      TAVG=0
      DELR=DELX*D/2
C++++++++++++++++++++++++++++++++++++++++++++++++++++++
      WNREL=SQRT(1.+1.5*EPS/(1.-EPS))
      DAMP=.5*(C0+CT)*RHO*5.73*(D/2.)**4/16./MIF*(1.-EPS)**4*
     &(1.+1./3.*EPS)/(1.-EPS)/WNREL
      CDELAY=(WNREL**2-1.)/SQRT((WNREL**2-1.)**2+4.*(DAMP*WNREL)**2)
      DELAY=PI/2.-ACOS(CDELAY)
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++
      PSI00=PSI0-KTEN*DELPSI
      PSI=PSI00
      OVER0=0.0
      THRST1=0.
2280  CONTINUE
      SPSI=SIN(PSI)
      CPSI=COS(PSI)
      SALPHA=SIN(ALPHA)
      CALPHA=COS(ALPHA)
C-------------------------------------------------------
C This block is the First-Order Hold
      THETA1=UTHET1(2)+(UTHET1(2)-UTHET1(1))/DELPSI*
     & (KTEN*DELPSI/(NPRED*2.*PI))*(PSI-PSI00)
```

```
      THETA2=UTHET2(2)+(UTHET2(2)-UTHET2(1))/DELPSI*
     & (KTEN*DELPSI/(NPRED*2.*PI))*(PSI-PSI00)
C-------------------------------------------------------
C
      LAMDA=MU*ALPHA
      DO 9050 I=1,20
      CTR=ASIG/2.*(LAMDA*T1+(THETA0+KBETA*BETA0)*T2+THETAT*T3+(THETA2
     1-KBETA*B1)*T4)
      CFAC=V**4+(TTAVG/RHO/AREA)**2+W**3*2.*V*SIN(A1+ALPHA)
      IF(CFAC.LT.0.) THEN
         W=SQRT(0.5*(-V**2+SQRT(V**4+(TTAVG/RHO/AREA)**2)))
         GO TO 9051
      ENDIF
      WFAC=-V**2+SQRT(CFAC)
      IF(WFAC.LT.0.) THEN
         W=SQRT(0.5*(-V**2+SQRT(V**4+(TTAVG/RHO/AREA)**2)))
         GO TO 9051
      ENDIF
      W=SQRT(0.5*(-V**2+SQRT(CFAC)))
9051  WVT=W/VT
      LAMDA=MU*ALPHA-WVT
      BETA0=GAMMAF*(LAMDA*F1+(THETA0+KBETA*BETA0)*F2+THETAT*F3+(THETA2-
     1KBETA*B1)*F4)-TAU
      A1=LAMDA*A11+(THETA0+KBETA*BETA0)*A12+THETAT*A13+(THETA2-KBETA*B1
     1)*A14
      B1=BETA0*B11-(THETA1-KBETA*A1)
9050  TTAVG=RHO*AREA*VT**2*CTR
      THRST2=0
C     START OF INTEGRATION OVER X FROM XH TO 1
      EPSR=EPS*D/2
      R=EPSR
      M1=0
      DT1=0
      MAERO=0
C     RETURN TO HERE AFTER INCREMENTING RADIUS, R
2380  X=R/D*2
      C=C0-(C0-CT)*X
      THETA=THETA0+THETAT*X+THETA1*CPSI+THETA2*SPSI+KBETA*BETA
      VTHETA=OMEGA*R+V*CALPHA*SPSI
      VU=V*SALPHA-W-(R-EPSR)*BETAD-BETA*V*CALPHA*CPSI
      ALPHAB=ATAN(ABS(VU/VTHETA))
      CLFAC=1.0
      CDFAC=1.0
      IF (VTHETA.GE.0.0.AND.VU.GE.0.0)THEN
           ALPHAB=THETA+ALPHAB
           IF(ALPHAB.LT.0.0)CLFAC=-1.
           ALPHAB=ABS(ALPHAB)
           GO TO 2381
      ENDIF
      IF (VTHETA.GT.0.0.AND.VU.LT.0.0) THEN
            ALPHAB=THETA-ALPHAB
            IF(ALPHAB.LT.0.0)CLFAC=-1.
            ALPHAB=ABS(ALPHAB)
            GO TO 2381
      ENDIF
      IF (VTHETA.LT.0.0.AND.VU.LT.0.0) THEN
            ALPHAB=PI-ALPHAB-THETA
            CLFAC=-1.0
            CDFAC=-1.0
            IF(ALPHAB.GT.PI)CLFAC=1.0
            IF(ALPHAB.GT.PI)ALPHAB=2.*PI-ALPHAB
            GO TO 2381
      ENDIF
      IF (VTHETA.LT.0.0.AND.VU.GT.0.0) THEN
             ALPHAB=PI+THETA-ALPHAB
             CLFAC=+1.0
             CDFAC=-1.0
             IF(ALPHAB.GT.PI)CLFAC=-1.0
             IF(ALPHAB.GT.PI)ALPHAB=2.*PI-ALPHAB
      ENDIF
2381  CALL AIRFOIL
      CL=CL*CLFAC
      CD=CD*CDFAC
      VRSQD=VU**2+VTHETA**2
      DLDR=RHO/2*VRSQD*CL*C
      DDDR=RHO/2*VRSQD*CD*C
      VR=SQRT(VRSQD)
      DT2=DLDR*ABS(VTHETA/VR)+DDDR*ABS(VU/VR)
      M2=(R-EPSR)*DT2
      DMDR=(M1+M2)/2
      DTDR=(DT1+DT2)/2
      M1=M2
      DT1=DT2
      MAERO=MAERO+DMDR*DELR
      THRST2=THRST2+DTDR*DELR
      R=R+DELR
      IF (R.GT.D/2) GO TO 2690
      GOTO 2380
2690  BETDD2=MAERO/MIF-BETA*OMEGA**2-MW/MIF
      BETAD2=BETAD2+(BETDD1+BETDD2)/2.*DELT
      BETAD=BETAD2
      BETA=BETA+(BETAD1+BETAD2)/2.*DELT
      BETDD1=BETDD2
      BETAD1=BETAD2
      M1=0.
      DT1=0.
      PSI2=PSI
      PSI1=PSI2
      THRUST=(THRST1+THRST2)/2.
      THRST1=THRST2
      TAVG=TAVG+THRUST*DELPSI
C
      BETLIM=-1.*DTR
      IF(BETA.LT.BETLIM) THEN
      OVER=BETA-BETLIM
      IF(ABS(OVER).LT.ABS(OVERO)) THEN
C  SOS IS THE ANGLE AT WHICH MAXIMUM OVERFLAPPING OCCURS
      SOS=PSI+DELAY-DELPSI-AINT((PSI+DELAY-DELPSI)/2./PI)*2.*PI
      BEEP='Y'
      U(10)=ABS(OVERO)
      DO 210 IU=1,9
210   U(IU)=UO(IU+1)
      CALL FDBACK(OVERO)
      CNTRL1=-GAIN*OVERO*SIN(SOS)/DTR
      CNTRL2=GAIN*OVERO*COS(SOS)/DTR
      WRITE(105,61) NREV,CNTRL1,CNTRL2,E1,E2
61    FORMAT(1X,5G15.8)
      GOTO 100
      ELSE
      OVERO=OVER
      ENDIF
      ELSE IF(BEEP.EQ.'Y'.AND.ABS(PSI-AINT(PSI/2./PI)*2.*PI-SOS)
     &.LT.DELPSI/1.9.AND.(PSI-PSIO).GT.4.*PI) THEN
      UNDER=BETA-BETLIM
C 0.2 IN THE FOLLOWING FORMULA IS THE FLAPPING ANGLE ERROR (IN DEGREES) ALLOWED
      IF((UNDER-.2*DTR).GT.0.) THEN
      U(10)=ABS(BETA)
      DO 220 IU=1,9
220   U(IU)=UO(IU+1)
      CALL FDBACK(UNDER-.2*DTR)
      CNTRL1=-GAIN*(UNDER-0.2*DTR)*SIN(SOS)/DTR
      CNTRL2=GAIN*(UNDER-0.2*DTR)*COS(SOS)/DTR
      WRITE(105,61) NREV,CNTRL1,CNTRL2,E1,E2
      SINGAL='O'
      GOTO 100
      ENDIF
      ENDIF
C+++++++++++++++++++++++++++++++++++++++++++++++++
C
2790  PSI=PSI+DELPSI
      PSII=PSII+ DELPSI
      IF (PSII.LE.2.*PI) GO TO 9801
      TAVG=TAVG/2./PI*B
      CTRW=TAVG/RHO/AREA/VT**2
      DO 9800 I=1,5
      CFAC=V**4+(TAVG/RHO/AREA)**2+W**3*2.*V*SIN(A1+ALPHA)
      IF(CFAC.LT.0.)THEN
         W=SQRT(0.5*(-V**2+SQRT(V**4+(TAVG/RHO/AREA)**2)))
         GO TO 9800
      ENDIF
      WFAC=-V**2+SQRT(CFAC)
      IF(WFAC.LT.0.)THEN
         W=SQRT(0.5*(-V**2+SQRT(V**4+(TAVG/RHO/AREA)**2)))
         GO TO 9800
      ENDIF
      W=SQRT(0.5*(-V**2+SQRT(CFAC)))
 9800 WVT=W/VT
      TAVG=0.
      PSII=0.
 9801 IF((PSI-PSIO).GT.NPRED*2.*PI) THEN
      BEEP='N'
      CNTRL1=0.
      CNTRL2=0.
      E1=THETA1/DTR
      E2=THETA2/DTR
      WRITE(105,61) NREV,CNTRL1,CNTRL2,E1,E2
      GOTO 100
      ENDIF
      GOTO 2280
100   WRITE(107,62) NREV,GAIN
62    FORMAT(1X,2G15.8)
      RETURN
      END
C
C------------------------------------------------------------------------
C This subprogram renews the GAIN so that the system stability may be
C guaranteed although the gain so obtained makes the settling time longer.
      SUBROUTINE FDBACK(OVER)
      REAL U(10),U0(10)
      CHARACTER*1 BEEP
      COMMON/PMT/PI,DTR,STEP,NMAX,PSI0,DELPSI,W0
      COMMON/INPUT/THETA0,THETA1,THETA2,ALPHA,BETA0,BETAD0
      COMMON/SIMU/B,D,C0,CT,EPS,KBETA,THETAT,MW,MIF,V,VT,DELX
     &,RHO,KTEN,CRATE0,CRATE1,CRATE2,CRATEA,GAIN,SOS,BEEP,U,U0
C
C An adaptive GAIN is chosen so as to guarantee the system stability.
C The design principle is to place more confidence on more recent controls
C with exponential confidence coefficients.
C ANORM(U) is a functional subprogram which gives:
C     ANORM(U)=[U(10)+exp(-c*1)*U(9)+...+exp(-c*9)*U(1)]/2-NORM OF U
C where c is the decaying coefficient.
C
C THE FOLLOWING IF-ENDIF BLOCK GIVES A SEMI-ADAPTIVE GAIN
      IF(ANORM(U).GE.ANORM(U0).AND.ANORM(U0).GT.1.E-4) THEN
      GAIN=ANORM(U0)/ANORM(U)*GAIN
      ENDIF
      IF(GAIN.LE.0.8) GAIN=0.8
C
      THETA1=THETA1-GAIN*OVER*SIN(SOS)
      THETA2=THETA2+GAIN*OVER*COS(SOS)
C The physical limits on THETA1 and THETA2 are given below:
      IF(THETA1.GE.15.*DTR) THETA1=15.*DTR
      IF(THETA1.LE.-15.*DTR) THETA1=-15.*DTR
      IF(THETA2.GE.15.*DTR) THETA2=15.*DTR
      IF(THETA2.LE.-15.*DTR) THETA2=-15.*DTR
      DO 10 I=1,10
10    U0(I)=U(I)
      RETURN
      END
C
C--------------------------------------------------------------------
C This subprogram defines a special kind of NORM for a vector used in
C subprogram FDBACK. The NORM so defined has the following properties:
C     1. NORM(U) is smaller than or equal to infinite-Norm of U
C     2. Each coefficient of U has different weighting on NORM(U),
C        which is different from the usual P-Norm.
```

```
C
      FUNCTION ANORM(U)
        REAL U(10)
        C=0.1
      ANORM=0.0
        DO 10 I=1,10
10      ANORM=ANORM+EXP(-C*FLOAT(10-I))*U(I)
      RETURN
      END
```

Input data files for FORTRAN programs
(AH-1J case)

FLP5.DAT FOR THE AH-1J

```
44.
2.25
2.25
.01
```

FLP6.DAT FOR THE AH-1J

```
0.
-10.
3122.
1422.
61.0
```

FLP7.DAT FOR THE AH-1J

```
738.
15.27
1.73
0.11
-4.48
2.71
-1.24
2.6
```

FLP8.DAT FOR THE AH-1J

```
.05
5.
.002378
9500.
```

CL.DAT FOR 0012 AIRFOIL

```
0.       0.
2.        .211
4.        .422
6.        .633
8.        .844
10.      1.055
11.      1.161
12.      1.255
13.      1.334
14.      1.333
15.      1.19
16.      1.007
21.       .800
39.      1.18
49.      1.18
129.     1.
147.     1.
161.      .62
172.      .78
180.      .0
```

CD.DAT FOR 0012 AIRFOIL

```
0.      .008
1.      .0083
2.      .0085
3.      .0088
4.      .0093
5.      .01
6.      .011
7.      .0122
8.      .0138
9.      .0154
10.     .0174
11.     .0196
12.     .022
13.     .0264
14.     .038
15.     .102
16.     .155
21.     .332
30.     .562
50.     1.392
60.     1.66
70.     1.84
80.     1.96
90.     2.02
100.    2.02
110.    1.852
120.    1.652
140.    1.042
160.    .302
165.    .242
170.    .132
175.    .062
180.    .022
```